**TITLUL LUCRĂRII DE LICENŢĂ**

LUCRARE DE LICENŢĂ

|  |  |  |
| --- | --- | --- |
|  | Absolvent: | **Prenume NUME** |
|  |  |  |
|  | Coordonator ştiinţific: | **titlul ştiinţific Prenume NUME** |

**2020**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | |  | |
| DECAN, |  | | DIRECTOR DEPARTAMENT, | |
| **Prof. dr. ing. Liviu MICLEA** |  | **Prof. dr. ing. Rodica POTOLEA** | |

Absolvent: **Prenumele NUMELE**

**TITLUL LUCRĂRII DE LICENŢĂ**

1. **Enunţul temei:** *Scurtă descriere a temei lucrării de licenţă şi datele inițiale*
2. **Conţinutul lucrării:** *(enumerarea părţilor componente) Exemplu: Pagina de prezentare, aprecierile coordonatorului de lucrare, titlul capitolului 1, titlul capitolului 2,… titlul capitolului n, bibliografie, anexe.*
3. **Locul documentării**: *Exemplu*: Universitatea Tehnică din Cluj-Napoca, Departamentul Calculatoare
4. **Consultanţi**:
5. **Data emiterii temei:** 1 noiembrie 2019
6. **Data predării:** 8 iulie 2020

|  |  |
| --- | --- |
| Absolvent: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |
| Coordonator ştiinţific: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

**Declaraţie pe proprie răspundere privind**

**autenticitatea lucrării de licenţă**

Subsemnatul(a)**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**, legitimat(ă) cu \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ seria \_\_\_\_\_\_\_ nr. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
CNP \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, autorul lucrării \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_elaborată în vederea susţinerii examenului de finalizare a studiilor de licență la Facultatea de Automatică și Calculatoare, Specializarea \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ din cadrul Universităţii Tehnice din Cluj-Napoca, sesiunea \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ a anului universitar \_\_\_\_\_\_\_\_\_\_, declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, pe baza cercetărilor mele şi pe baza informaţiilor obţinute din surse care au fost citate, în textul lucrării, şi în bibliografie.

Declar, că această lucrare nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în faţa unei alte comisii de examen de licenţă.

In cazul constatării ulterioare a unor declaraţii false, voi suporta sancţiunile administrative, respectiv, *anularea examenului de licenţă*.

|  |  |  |
| --- | --- | --- |
| Data  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  | Nume, Prenume  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |  |
|  |  | Semnătura |

**De citit înainte** (această pagină se va elimina din versiunea finală):

1. Cele trei pagini anterioare (foaie de capăt, foaie sumar, declaraţie) se vor lista pe foi separate (nu faţă-verso), fiind incluse în lucrarea listată. Foaia de sumar (a doua) necesită semnătura absolventului, respectiv a coordonatorului. Pe declaraţie se trece data când se predă lucrarea la secretarii de comisie.
2. Pe foaia de capăt, se va trece corect titulatura cadrului didactic îndrumător (consultaţi pagina de unde aţi descărcat acest document pentru lista cadrelor didactice cu titulaturile lor).
3. Documentul curent a fost creat în **MS Office 2007.** Dacă folosiţi alte versiuni e posibil sa fie mici diferenţe de formatare, care se corectează (textul conţine descrieri privind fonturi, dimensiuni etc.).
4. **Cuprinsul** începe pe pagina nouă, impară (dacă se face listare faţă-verso), prima pagina din capitolul **Introducere** tot aşa, fiind numerotată cu 1. Pentru actualizarea cuprinsului, click dreapta pe cuprins (zona cuprinsului va apare cu gri), Update field->Update entire table.
5. Vizualizaţi (recomandabil şi în timpul editării) acest document după ce activaţi vizualizarea simbolurilor ascunse de formatare (apăsaţi simbolul **** din *Home/Paragraph*).
6. Fiecare capitol începe pe pagină nouă, datorită simbolului ascuns Section Break (Next Page) care este deja introdus la capitolul precedent. Dacă ştergeţi din greşeală simbolul, se reintroduce (*Page Layout -> Breaks*).
7. Folosiţi stilurile predefinite (Headings, Figura, Tabel, Normal, etc.)
8. Marginile la pagini nu se modifică (Office 2003 default).
9. Respectaţi restul instrucţiunilor din fiecare capitol.

Cuprins

[Capitolul 1. Introducere – Contextul proiectului (Heading 1 style) 1](#_Toc384994105)

[1.1. Contextul proiectului (Heading 2 style) 1](#_Toc384994106)

[1.1.1. (Heading 3 style) 1](#_Toc384994107)

[Capitolul 2. Obiectivele proiectului 3](#_Toc384994108)

[Capitolul 3. Studiu bibliografic 4](#_Toc384994109)

[Capitolul 4. Analiză şi fundamentare teoretică 5](#_Toc384994110)

[Capitolul 5. Proiectare de detaliu si implementare 5](#_Toc384994111)

[Capitolul 6. Testare şi validare 5](#_Toc384994112)

[Capitolul 7. Manual de instalare si utilizare 5](#_Toc384994113)

[Capitolul 8. Concluzii 5](#_Toc384994114)

[Bibliografie 5](#_Toc384994115)

[Anexa 1 (dacă este necesar) 5](#_Toc384994116)

# Introducere

Acum aproximativ 7.000 - 10.000 de ani, undeva în Semiluna fertilă (zonă din vestul Asiei, parte care include Mesopotamia și Levantul) homo sapiens a început agricultura de subzistență. Dacă până atunci oamenii erau obișnuiți să își ia sursa de hrană din ce voia natura să le ofere, acum, omul începe să controleze natura astfel încât aceasta să îi ofere ce vrea el. Acum, omul a început să își pregătească pământul, să semene grâul, să îl crească, îngrijească, să îl ferească de prădători, să îl recolteze, depoziteze, ca mai apoi să îl consume.

După 10.000 de ani, agricultura, întregul proces de creștere a plantelor, este de nerecunoscut. Deși în esență e vorba de parcurgerea acelorași etape, modul și viteza cu care acestea sunt parcurse în ziua de azi este cu totul diferit. În ultimele sute de ani agricultura a evoluat substanțial. Cea mai de ne recunoscut schimbare fiind automatizarea acesteia. Serele folosesc sisteme automate de monitorizare și calibrare a temperaturii, umidității din aer, umidității solului, a lumini, etc pentru a se asigura că obțin cea recolta cea mai bună în timpul cel mai scurt și cu cele mai mici costuri.

Ingineri și botaniști specializați proiectează sere ce caută în continu dobândirea celei mai ample recolte cu cel mai redus buget. Unele dintre cele mai ostentative alterări aduse procesului de creștere a plantelor  sunt verticalizarea culturilor, înlocuirea solului cu apă și folosirea luminii artificiale de creștere. Acestea sunt și modificările care aduc cel mai mult profit. Prin agricultura pe verticală, aceeași bucată de pământ poate produce de 4 – 5 sau chiar 10 ori mai mult. Înlocuirea solului cu apă face posibilă dezvoltarea recoltei indiferent de zona geografică și resursele existente. Plasarea de lumini artificiale (la distanțe precis calculate de fiecare plantă) asigură dezvoltarea plantelor în cel mai rapid mod. Alte modificări aduse procesului de cultivare a plantelor sunt controlul temperaturii aerului sau controlul umidității aerului sau al solului. Acestea sunt posibile datorită segmentului de IOT (Internet of Things) .

Noțiunea de IOT a evoluat și și-a schimbat definiția de-a lungul anilor dar în ultimii ani am putut să ajungem cel puțin la consensul că Internet of Things este un sistem care interconectează dispozitive de calcul cu mașinării mecanice și/sau digitale identificabile în mod unic, care are abilitatea să transfere informație în cadrul rețelei fără a necesita interacțiuni între oameni sau între om și calculator.

La scară largă, pentru practicarea agriculturii, am văzut că există soluții tehnice și proiectanți specializați care pot implementa aceste soluții.

La scară mică, putem vorbi de agricultura de subzistență iar la scară micro, de creșterea plantelor pentru decor. În ambele cazuri putem spune că în majoritatea timpului nu avem proiectanți specializați și nici de tehnologii IOT care să optimizeze procesul de creșterea a plantelor. Pentru prima lipsă, cea a personalului specializat, nu putem veni cu o soluție ușoară, dar pentru cea de-a doua, soluțiile încep să apară la prețuri tot mai accesibile. În continuare ne vom axa pe creșterea plantelor pentru decor, mai exact, pe creștea plantelor de ghiveci și cum poate tehnologia IOT să facă această slujbă mai ușoară și mai sigură.

## Contextul proiectului

Creșterea plantelor pentru decor, în special a platelor de ghiveci în apartamente este influențată de mai mulți factori, câțiva dintre aceștia meritând a fi amintiți cum ar fi luminozitatea de care beneficiază planta, tipul solului în care aceasta este plantată, dimensiunea ghiveciului comparativ cu dimensiunea rădăcinilor și cel mai important, udarea corespunzătoare. În timp ce primii trei factori pot fi corectați într-o acțiune singulară, asigurarea cantității optime de apă pentru plante necesită un efort constant. Udarea deficitară a acestora este cauza principală pentru care plantele de apartament se ofilesc și mor.

În salvarea vegetație de apartament vin sistemele inteligente din lumea IOT care oferă o varietate de soluții, începând de la notificări pentru utilizator ca acesta să își ude plantele, până la automatizarea completă a procesului de creștere a plantelor.

Un sistem IOT este format atât din componente hardware cât și software. Pentru a oferi o imagine de ansamblu asupra lumii IOT, atât am ales să o împart în patru subcategorii ca apoi să evidențiez rolul și compoziția fiecăreia. După cum se observă și în figura 1.1 categoriile sunt: Dispozitivele de măsurare, Dispozitivele acționabile, Logica de control (împreună cu dispozitivele de stocare a acesteia) și Interfața cu utilizatorul (pentru care sunt necesare și dispozitive cu care utilizatorul se poate conecta la respectiva interfață).

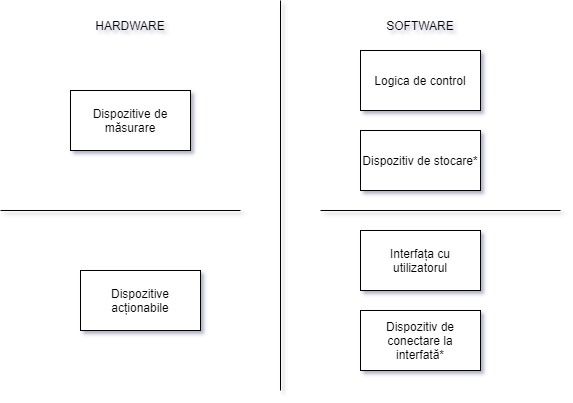
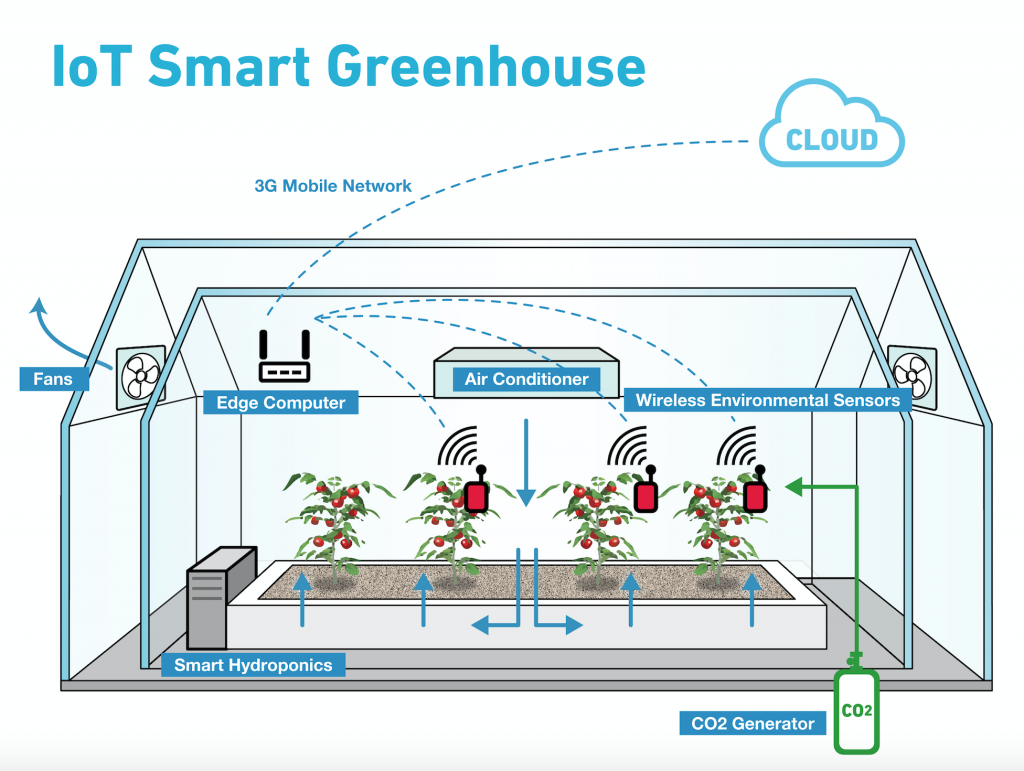


Figura . Componente ale unui sitem IOT

Prima componentă hardware sunt dispozitivele de măsurare și ele se referă la orice dispozitiv ce poate lua informație din mediu și o poate converti în informație analogică sau digitală ce poate fi interpretată de logica de control. Exemple populare de dispozitive de măsurare ar fi senzorii sau camerele de luat imagini. În majoritaea cazurilor aceste dispozitive sunt capabile să culeagă și transmită informația mai departe în regim autonom, fără interacțiunea unui utilizator. Aceasta este și trăsătura principală a unui sistem IOT, faptul că poate funcționa în mod independent de interacțiune umană, de aceasta fiind nevoie doar pentru instalare și calibrare.

A doua componentă hardware e reprezentată de dispozitivele acționabile. În cadrul unui sistem IOT, ele sunt acele dispozitive care sunt controlate de sistem și care pot aduce modificări mediului. Exemple reprezentative ar fi sistemele de iluminat, pompele de apă, generatoarele de căldură, etc. Acestea sunt acționate de logica de control.

Logica de control dintr-un sistem IOT este partea software care se poate ocupa cu recepționarea informației de la dispozitivele de măsurare, recepționarea comenzilor de la utilizator, procesarea acestor date și/sau comenzi și controlul dispozitivelor acționabile în funcție de rezultatul acestei procesări. Logica de control poate fi centralizată pentru întreg sistemul și stocată în cloud sau pe un server în rețea, dar ea poate fi de asemenea și decentralizată, fiecare dispozitiv venind cu propria logică de control. În exemplul din figura 1.2 observăm o seră inteligentă în care logica de control e stocată în cloud, agregă date de la mai mulți senzori – „Wireless Envionmental Sensors” – și controlează mai multe dispozitive – „Fans”, „Smart Hydroponics”, „CO2 Generator”, „Air Conditioning”) – ce pot altera starea mediului. În acest caz nu e nevoie de interacțiunea umană.

Figura . Seră inteligentă[[1]](#footnote-2)

Spre deosebire de figura 1.2 în figura 1.3 avem un concept de sistem IOT în care logica de control e stocată direct pe dispozitivul IOT iar interacțiunea utilizatorului e necesară pentru ca acest sistem IOT să aibe sens. Mai precis, în acest caz avem un ceainic inteliget ce dispune de senzor de temperatură a apei și care poate fi controlat de la distanță prin intermediul unui interfețe accesibilă cu un smartphone. Utilizatorul poate motifica mediul (temperatura apei din ceainic) prin plasarea unei comenzi în cadrul interfeței, care la rândul ei va comanda dispozitivul acționabil – ceainicul – să încălzească apa până la o anumită temperatură.

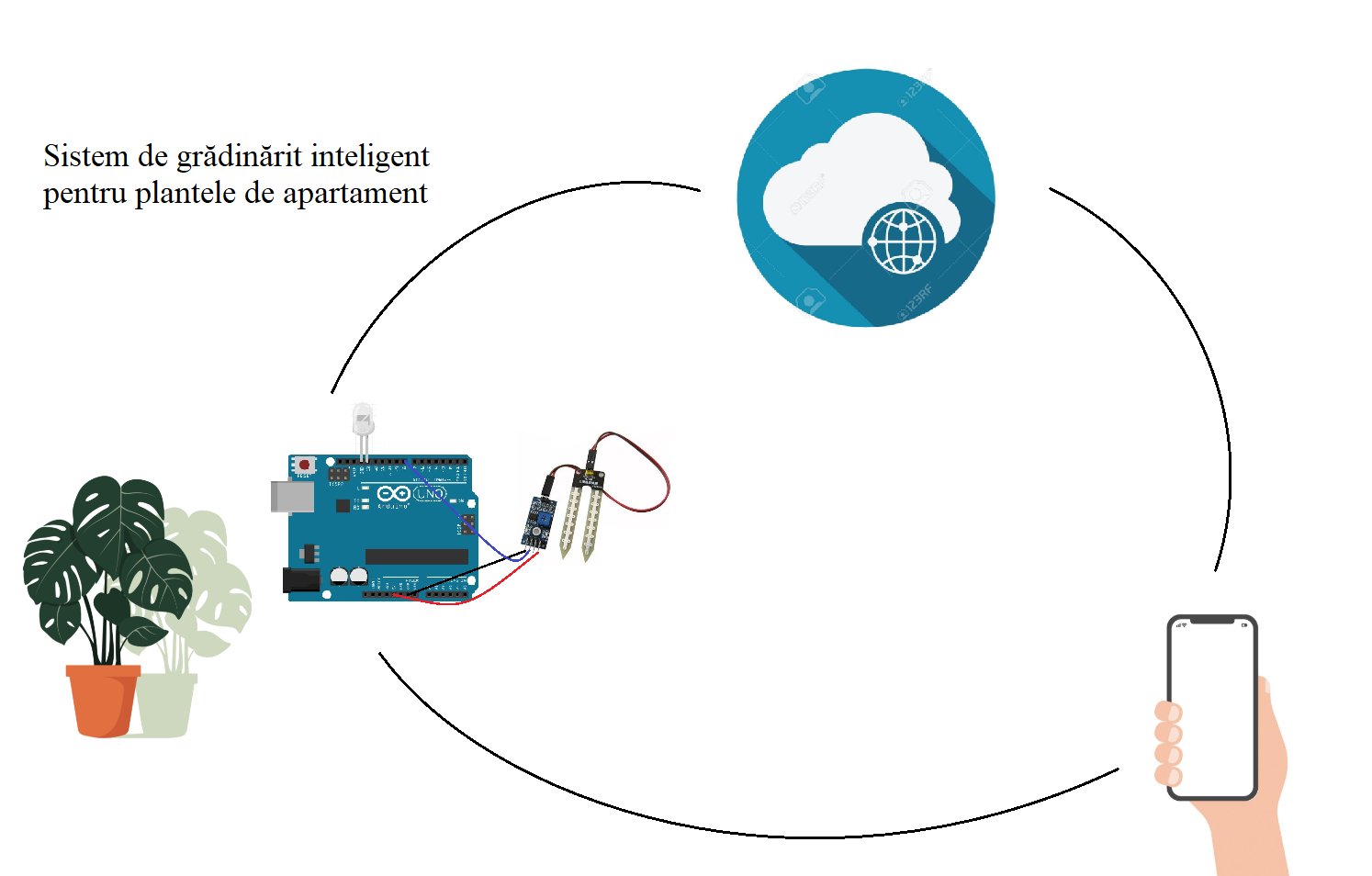
Figura . Sistem IOT cu ceainic inteligent[[2]](#footnote-3)

Utilma subcategorie pe care o prezentăm în cadrul unu sistem IOT este interfața pentru utilizator. Aceasta e deseori reprezentată de o aplicație web sau de o aplicație mobile care are ca scop facilitarea interacțiuni dintre utilizator și sistemul inteligent. Funcțiile pe care interfața le oferă sunt, de regulă, funcții de monitorizare și opțional de controlare a mediului sistemului. În exemplele noastre funcțiile de monitorizare ar putea oferi informații cu privire la nivelul de CO2 din seră sau la temperatura apei din ceainic, în timp ce funcțiile de alterare a mediului ar putea modifica nivelul de CO2 din seră sau temperatura apei din ceainic.

# Obiectivele Proiectului

Lucrarea de față își propune proiectarea unui sistem inteligent de grădinărit de tipul IOT pentru monitorizarea și automatizarea parțială a creșterii plantelor de apartament. În figura 2.1 este ilustrat succint compoziția unui asfel de sistem de grădinărit. El se compune din:

* Componente hardware de măsurare și alterare a mediului.
* Aplicație de tipul server pentru stocare de date
* Aplicație de tip client pentru interacțiunea utilizatorului cu sistemul

Figura . Sistem de grădinărit inteligent

Scopul principal ce se dorește atins este acela de a oferi un prototip care poate să crească diverse plante în ghiveci cu interacțiune minimă din partea utilizatorului. Scopul e de a crește plantele dintr-un singur ghiveci cu un sistem de grădinărit și de a monitoriza creșterea acestora. Scopul secundar este a pune la dispoziție utilizatorilor care nu au acces la partea de hardware a sistemului de grădinărit inteligent o aplicatie care facilitiază creșterea plantelor.

Lucrarea de față va încerca să își atingă obiectivele prin următoarele:

* Monitorizarea mediului
* Controlul mediului
* Stocarea și deservirea de infomații despre plante și la mediul acestora
* Oferirea unei interfețe web prin care utilizatorul poate comunica cu sistemul

.

## Cerințe funcționale

Pentru a-și atinge obiectivele, sistemul oferă funcționalități care implică interacțiunea utilizatorului pe tot procesul scenariului, cât și funcționalități care implică interacțiunea limitată a utilizatorului cu sistemul pe parcusul unui scenariu. Funcționalitățile care implică interacțiunea utilizatorului pe tot parcursul scenariului de utilizare sunt acelea care au ca scop stocare de informații din partea utilizatorului și deservirea acestuia cu informații. Ele sunt:

* Autentificarea
* Navigarea prin lista plantelor stocate în sistem
* Crearea unei grădini virtuale cu plantele personale ale utilizatorului
* Adăugarea de plante noi care nu sunt în sistem în grădina virtuală
* Vizualizarea nevoilor unei plante (stocate în sistem sau create de utilizator)
* Afișarea datelor despre mediul plantelor din grădină

Scenariile ce implică o interacțiune limitată a utilizatorului cu sistemul au în vedere creșterea eficientă a plantelor. Acestea sunt scenariile care necesită partea de harware a sistemului de grădinărit. Ele sunt:

* Monitorizarea mediului
* Păstrarea umidității soului adecvată

### Autentificarea

Funcționalitatea de autentificare permite utilizatorului să se logheze în cadrul sistemului și să salveze date despre plantele sale. Aceste date sunt persistate și sunt accesibile doar utilizatorului în cauză.

### Navigarea prin lista plantelor stocate în sistem

Sistemul beneficiază de o bază de date cu cele mai întâlnite plante de apartament din zona geografică a României. Utilizatorii pot naviga prin această listă de plante pentru a identifica planta pe care vor să o crească cu ajutorul sistemului de grădinărit ineltigent sau cu scopul de a-și alege o plantă în funcție de nevoile acesteia și mediul pe care i-l pot oferi.

### Crearea unei grădini virtuale

Pentru a ține o mai bună evidență a plantelor din apartament, utilizatorul își poate crea o grădină virtuală în care să stocheze plantele pe care le deține. De notat că pentru a monitoriza mai multe ghivece, e nevoie de mai multe prototipuri de sisteme inteligente de grădinărit (un sistem de grădinărit, întreține doar plantele dintr-un singur ghiveci). Această funcționalitate de creare a unei grădini virtuale oferă posibilitatea utilizatorului să țină o evidență a frecvenței de udare și a plantelor care nu beneficiază de un sistem de grădinărit inteligent.

### Adăugare de plante noi în grădina virtuală

Întrucât sistemul nu deține toate plantele existente, această opțiune de a adăgua alte plante oferă utilizatorului posibilitatea să stocheze în grădina sa virtuală o plantă pe care să o descrie manual. A descrie manual o plantă înseamă în acest caz descrierea nevoilor plantei respective. Nevoile de care ține cont sistemul sunt: temperatura aerului, umidiatea aerului, nivelul de luminozitate, nivelul de umiditate a solului, tipul solului, cantitatea de sol necesară unei plante.

### Vizualizarea nevoilor unei plante

Pentru a avea acces facil și compact la cât mai multe informații esențiale în vederea creșterii plantelor, fiecare plantă oferă informații cu privire la condițiile optime de dezvoltare. Informațiile care pot fi vizualizate pentru fiecare plantă sunt: temperatura aerului, umidiatea aerului, nivelul de luminozitate, nivelul de umiditate a solului, tipul solului, cantitatea de sol necesară.

### Afișarea datelor despre plantele din grădina virtuală

Acest scenariu deservește două posbilități. În cazul în care utilizatorul nu are un sistem inteligent de grădinărit, ci doar acces la aplicația web, acesta poate vedea frecvența cu care a udat planta. În cazut în care utilizatorul are acces și la sistemul de grădinărit inteligent și acesta este configurat, pentru planta respectivă utilizatorul poate vedea următoarela informații:

* Nivelul de umiditate al aerului
* Nivelul de umiditate al solului
* Temperatura aerului
* Intensitatea luminoasă a mediului

### Monitorizarea mediului

În cadrul acestui scenariu sistemul IOT monitorizează prin intermdiul unor senzori următoarele caracteristici ale mediului plantei:

* Nivelul de umiditate al aerului
* Nivelul de umiditate al solului
* Temperatura aerului
* Intensitatea luminoasă a mediului

În cazul în care una dintre măsurători nu este în parametri optimi, utilizatorul este înștiințat de aceasta la următoarea deschidere a aplicației.

De meționat pe scurt că pentru a avea acces la scenariul de monitorizare este nevoie înainte de completarea unui scenariu auxiliar de configurare a părții hardware a sistemului (scenariul de configurare implică accesul sistemului harware la rețeaua wi-fi a utilizatorului și detectarea acestuia prin interfața utilizator). Având în vedere că vorbim de un prototip în cazul de față, acest scenariu premergător va fi făcut de către dezvoltatorul prototipului întrucât implică și alterarea codului.

### Păstrarea umidității solului adecvată

O dată configurat, sistemul IOT va păstra umiditatea solului în parametri specificați. Pentru îndeplinirea cu succes a acestui scenariu în momentul în care sistemul rămâne fără apă va informa utilizatorul iar acesta trebuie să acționeze în concordantă, adică să reumple rezervorul de apă al sistemului.

## Cerințe non-funcționale

Pe partea de cerințe non-funcționale sunt atinse următoarele:

* Usability/Utilizabilitatea – interfața web va fi ușor și plăcut de folosit
* Acsessibility/Accesabilitatea – partea de software a proictului va fi accesibilă publicului larg
* Open source/Acces la codul sursă – pentru a încuraja comunitatea sa dezvolte cât mai multe proiecte IOT
* Documentația – astfel încât sistemul să fie ușor de utilizat

# Studiu Bibliografic

În lucrarea „Automation using IOT in greenhouse environment„ [9] autorii prezintă prototipul unui sistem care monitorizează și controlează diferite aspecte ale unei sere. Ei colecționează informații despre starea mediului din seră folosind senzori. Senzorii măsoară temperatura, intensitatea luminoasă, umiditatea aerului și umiditatea solului. Pentru o bună dezvoltarea a plantelor, autorii folosesc diferite praguri între care păstrează valorile temperaturi și a umidității în seră. Împreună cu o plăcuță de dezvoltare Neduino 3 Wi-Fi, aceștia reglează temperatura cu ajutorul unor ventilatoare acționate automat și cu ajutorul unor motare care acționează paravane solare în seră pentru. În această lucrare este implementată și o interfață către utilizator în care acesta este notificat de starea serei (măsurătorile înregistrate de senzori).

S. Aishwarya et al [10] prezintă o soluție de monitorizare a creșterii plantelor de asemenea luând în considerarea temperatura, umiditatea, intensitatea luminoasă dar informația este colectată de la senzori de către o plăcuță Arduino UNO. O funcționalitate în plus pe care o aduce [10] este monitorizarea stării plantei în sine, nu doar a datelor din seră. Autorii folosesc procesarea de imagini pentru a identifica dacă plantele se dezvoltă adecvat. Deși sera este controlată, există anumite insecte care pot dăuna plantei sau solul poate avea lipsuri de anumite minerale. În ambele cazuri frunzele plantelor vor evidenția această problemă prin anumite comportamente. De exemplu o carență de fier duce la îngălbenirea funzelor dar nu și a nervurilor[[3]](#footnote-4). Pentru a putea monitoriza astfel de scenarii autorii folosesc o cameră de luat vederi și o plăcuță Rasperry Pi care rulează un algoritm de procesarea a imaginilor ce identifică efectul a diveși dăunători.

Pentru a structura mai bine rezultatul studiului bibliografic în lucrarea de față, vom împărți această secțiune în două mari capitole: partea de proiectarea hardware și partea de proiectarea software. Partea de proiectare hardware constă în componentele fizice care măsoară starea mediului, transmit date, recepționează comenzi și alterează starea mediului supravegheat. Partea de dezvoltarea software face referire la aplicațiile ce furnizează restul funcționalităților menționate anterior. În continuare vom prezenta câteva din opțiunile existente pentru implementarea atât a părții hardware cât și software. De asemenea, tot în acest capitol, vom justifica alegerile făcute.

## Proiectarea hardware

În timpul documentării asupra situației actuale din domeniul grădinăritului smart am observat o abordare similară în proiectarea serelor smart, și anume prototipizarea. Majoritatea lucrărilor prezintă un prototip care poate fi extins să funcționeze la scară largă într-o seră de mari dimensiuni. Pentru partea hardware, cea mai importantă decizie este alegerea plăcuței de dezvoltare care va controla mediul. Secundară acesteia este alegerea senzorilor pe care îi vom folosi și a dispozitivelor care controlează mediul (motoare, pompe, ventilatoare, etc). În continuare vom prezenta opțiunile de dispozitive hardware atât pentru recepționarea datelor și logica de control cât și opțiunile pentru motitorizarea și alterarea mediului.

### Placa de dezvoltare

În continuare vom prezenta procesul prin care am trecut în alegerea unei plăci de dezvoltare adecvată nevoilor proiectului. S-a dorit ca placa de dezvoltare să poată să citească date de la senzori analogici, să fie ușor de alimentat de la o baterie, să fie fiabilă, să poată fi conectată la internet și să nu fie costisitoare din punct de vedere financiar. De asemenea, un aspect care nu a fost neglijat a fost prețul de achiziție. Întrucât lucrarea de față are ca scop creșterea unei plante de apartament în ghiveci, sistemul trebuie să își păstreze costurile într-o limită atractivă pentru utilizator.

Pentru a identifica ce opțiuni există când vorbim de placă de dezvoltare în domeniul serelor smart am recurs la lucrări în domeniu din ultimii zece ani. În timp ce în „An FPGA Based Computer System for Greenhouse Control” [12] autorii folosesc o plăcuță FPGA pentru achiziționarea de date de la senzori și controlul unor paravane în scopul protejării plantelor, [10] folosește o plăcuță Arduino Uno pentru recepționarea datelor și controlul unor ventilatoare și stropitori.

Anterior am menționat și placa de dezvoltare Netduino 3 Wi-Fi care e utilizată în [9]. O altă soluție posibilă este utilizarea unui Raspberry Pi după cum am observat în lucrarea”IOT Based Smart Greenhouse” [13] în care autorii o folosesc pentru recepționarea datelor și transmiterea lor direct către cloud.

Urmează o scurtă prezentare a fiecărei din plăcuțele mai sus menționate pentru a crea o imagine de ansamblu.

Raspberry Pi prezentat în figura 3.1 este un minicopmuter pe care rulează sistemul de operare Raspbian. El poate fi ușor conectat la internet atât prin Wi-Fi cât și prin Ethernet. Pentru comunicarea cu senzori, Raspberrry Pi are nevoie de anumite librării și/sau programe software. Deși nu are spațiu de stocare, el vine cu un slot pentru un card SD.



Figura . Raspberry Pi 3[[4]](#footnote-5)

Arduino UNO, prezentat în figura 3.2 e un microcontroler programabil. Limbajul de programare Arduino, specific acestei plăci de dezvoltare, este bazat pe C/C++. Arduino Uno nu poate fi conectată direct la internet ci are nevoie de alte componente hardware (de ex, Wi-Fi shield) pentru a dispune de această funcționalitate. Citirea datelor de la senzori se face foarte ușor cu acesta datorită interfeței simple și pinilor analogici. Arduino UNO vine direct cu spațiu de stocare integrat[[5]](#footnote-6).



Figura . Arduino UNO[[6]](#footnote-7)

Neduino este o platformă open-source bazată pe framework-ul .NET micro. Netduino 3 WiFi prezentat în figura 3.3 este bazat pe microcontrollerul Cortex-M4. El este foarte similar în materie de proiectare cu Arduion, fiind compatibile shiled-uri Arduino cu această plăcuță. WiFi-ul integrat face conectarea la internet foarte ușoară iar framwork-ul micro .NET facilitează dezvoltarea programelor cu această placă.

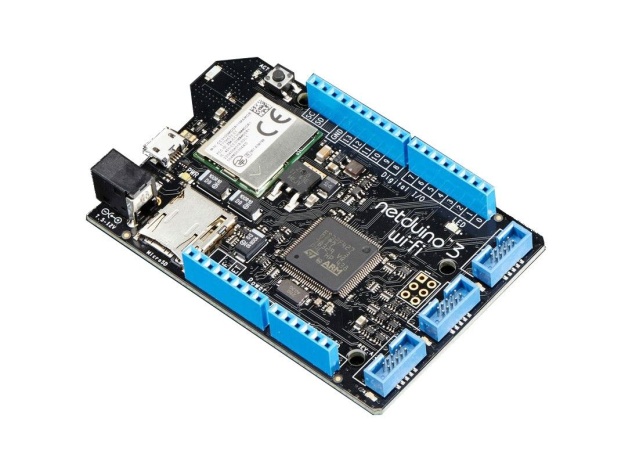


Figura . Netduino 3 WiFi[[7]](#footnote-8)

Plăcuța FPGA (Field Programable Gate Array) prezentată în figura 3.4 este un circuit integrat destinat pentru a fi configurare de către utilizator. Limbajul specific de programare a aceste plăci este HDL. Pentru conexiunea la internet aceasta necesită atât alte piese hardware cât și software specific. Citirea datelor de la senzori implică și ea mai multă configurare a plăcuței.

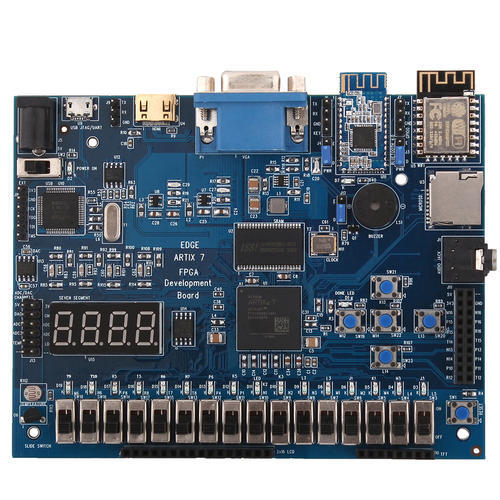


Figura . Plăcuță FPGA[[8]](#footnote-9)

Din opțiunile studiate și prezentate până acum, alegerea mea a fost plăcuță Netduino 3 WiFi dar în timpul căutării acesteia de la furnizori, am descoperit o alternativă recentă a elementului de control, și anune plăcuța UNO + Wi-Fi R3 care conține o integrare completă a microcontrolerului Atmel ATmega328 și a Wi-Fi-ului ESP8266 – figura 3.5. Ea este programabilă prin Arduino, are o interfață ușor de utilizat în interacțiunea cu elemetele perifierice și bineînțeles poate fi conectată cu ușurință la internet.

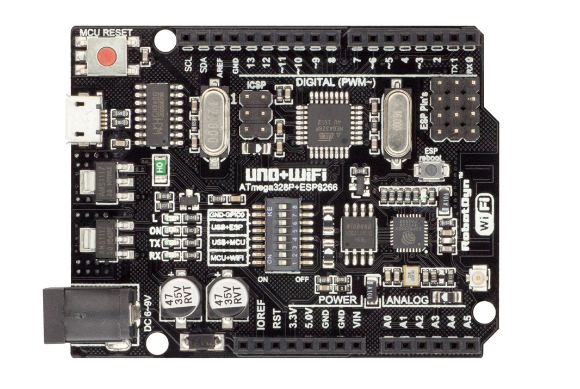


Figura . UNO + Wi-Fi R3[[9]](#footnote-10)

Pentru alegerea plăcuței cu care să proiectăm sistemul de grădinărit înteligent am evidențiat în tabelul 3.1 nivelul de dificultate cu care o putem programa, cât de ușor putem citi date de la senzori, lejeritatea cu care ne putem conecta la internet cât și prețul de achiziție.

Tabel . Comparare între dispozitivele hardware de control

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Plăcuță | Ușor și rapid programabil | Poate fi conecta la intenet fără alte resurse | Poate citi date direct de la senzori analogici | Preț de achiziție scăzut |
| Raspbery Pi | Da | Da | Nu | Nu[[10]](#footnote-11) |
| Arduino | Da | Nu | Da | Da[[11]](#footnote-12) |
| Netduino | Da | Da | Da | Nu[[12]](#footnote-13) |
| FPGA | Nu | Nu | Nu | Nu[[13]](#footnote-14) |
| Uno + Wi-Fi R3 | Da | Da | Da | Da[[14]](#footnote-15) |

Observăm că Uno + Wi-Fi R3 este singura opțiune care răspunde cu da cerințelor noastre. Fiind o opțiune cu modul de Wi-Fi integrat, aceasta reduce și posibilitate defectării componentelor auxiliare cum ar fi în cazul Arduino Uno. Este cu mult mai ieftină decât Raspbery Pi și de asemenea citirea datelor de la senzorii analogici se face cu o mai mare ușurință. Având în vedere scopul lucrării, microcontrolerul Uno + Wi-Fi R3 este cea mai potrivită alegere.

### Dispozitive periferice

În alegerea dispozitivelor perifcerice ne referim atât la cele care măsoara date desre mediu cât și la cele care alterează mediul.

Dispozitivele care măsoară date despre mediu sunt senzorii. Pentru a alege senzorii potriviți vom lua în considerare tipul de informație pe care aceștia o furnizează, modul de transmitere a datelor, și prețul de achiziție. În majoritatea lucrărilor studiate [9], [10], [11], [13] senzorii predominanți sunt:

* Senzor de temperatură a aerului
* Senzor te umiditate a aerului
* Senzor te umiditate a solului
* Senzor de intensitate luminoasă sau fotorezistor

Având în vedere că datele furnizate de acești senzori sunt suficiente pentru monitorizarea unei sere, am considerat că acești senzori pot măsura starea de bine a plantei și în cazul lucrării de față. Deoare niciunul dintre aceștia nu are un preț de achiziție ridicat [[15]](#footnote-16) [[16]](#footnote-17) [[17]](#footnote-18) am decis să îi folosim pe toți în lucrarea de față.

Dispozitivele care controlează mediul folosite în lucrările prezentate anterior sunt de tipul

* Ventilatoarea care reglează fluxul de aer
* Motoare care deplasează paravane solare (draperii)
* Pompe de apă pentru irigare.

Întrucât lucrarea de față își propune creșterea unei plante în ghiveci, nu în seră, modificare stării întregului apartament în care se află ghiveciul nu e fezabilă așa încât singurul dispozitv de alterare a mediului pe care îl vom folosi dintre acestea este pompa de apă.

## Proiectare software

Partea softawe a lucrării va deservi atât utilizatorii care dețin și partea hardware dar și pe cei care vor să utilizeze doar parte software. Pentru a servi nevoile utilizatorilor am ales să analizăm aplicații existente pe piață și nu lucrări științifice. Motivul acestei alegeri este că lucrările științifice nu au un grup de utilizatori larg care își poate spune părerea despre cât gradul de utilitate al proiectului, în timp ce aplicațiile existene dețin nenumărate păreri și recomandări din parte utilizatorilor.

Prima categorie de aplicații pe care am studiat-o este cea de identificare a plantelor. Aceastea sunt folositoare în cazul în care utilizatorul nu știe ce plantă deține și nu cunoaște nici nevoile acesteia. Aplicație pe care o prezentăm este „Garden Answers Plant Identification”[[18]](#footnote-19). Este vorba de o aplicație gratuită a cărei funcționalitate este de a identifica ce fel de plantă deține utilizatorul. Modul de utilizare al aplicației este următorul: utilizatorul face o poză plantei sale, o încarcă în aplicație iar aceasta sugerează o listă de opțiuni pe care le consideră adecvate. Utilizatorul trebuie să selecteze rezultatul care se potrivește cel mai mult cu planta sa. Ultima parte a acestui scenariu se poate obsera în figura 3.6.

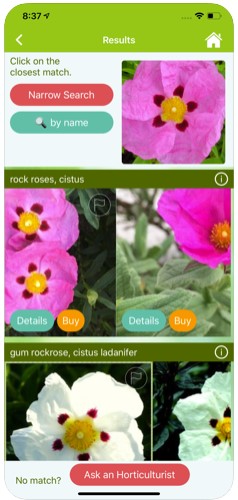


Figura . Garden Answers Plant Id – selectarea plantei

Cea de a doua categorie de aplicații din cele studiate pe care am ales să le observăm mai în detaliu este cea de păstrare a evidenței plantelor și regularitatea cu care acestea sunt udate. În această categorie intră aplicații precum „Waterbot: Plants watering + Gardening”[[19]](#footnote-20) și „Vera: Plant Care App”[[20]](#footnote-21). Cea din urmă, de asemenea gratuită ca și „Garden Answers Plant Identification”[[21]](#footnote-22), se axează pe udarea regulată a plantelor. Ea oferă posibilitatea utilizatorului să își creeze câte un profil pentru fiecare plantă pe care o deține – se poate observa în figura 3.7 – și să să seteze pentru fiecare plantă un program de udare în funcție de nevoile acesteia. Astfel, utilizatorul va avea un portofoliu de plante și va primi, separat, pentru fiecare plantă, o notificare în ziua în care aceasta trebuie udată. Aplicația ajută utilizatorul atât să își mențină plantele în viață prin udarea lor regulată cât și oferă o privire de ansamblu asupra plantelor pe care acesta le deține.

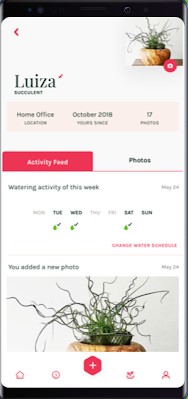
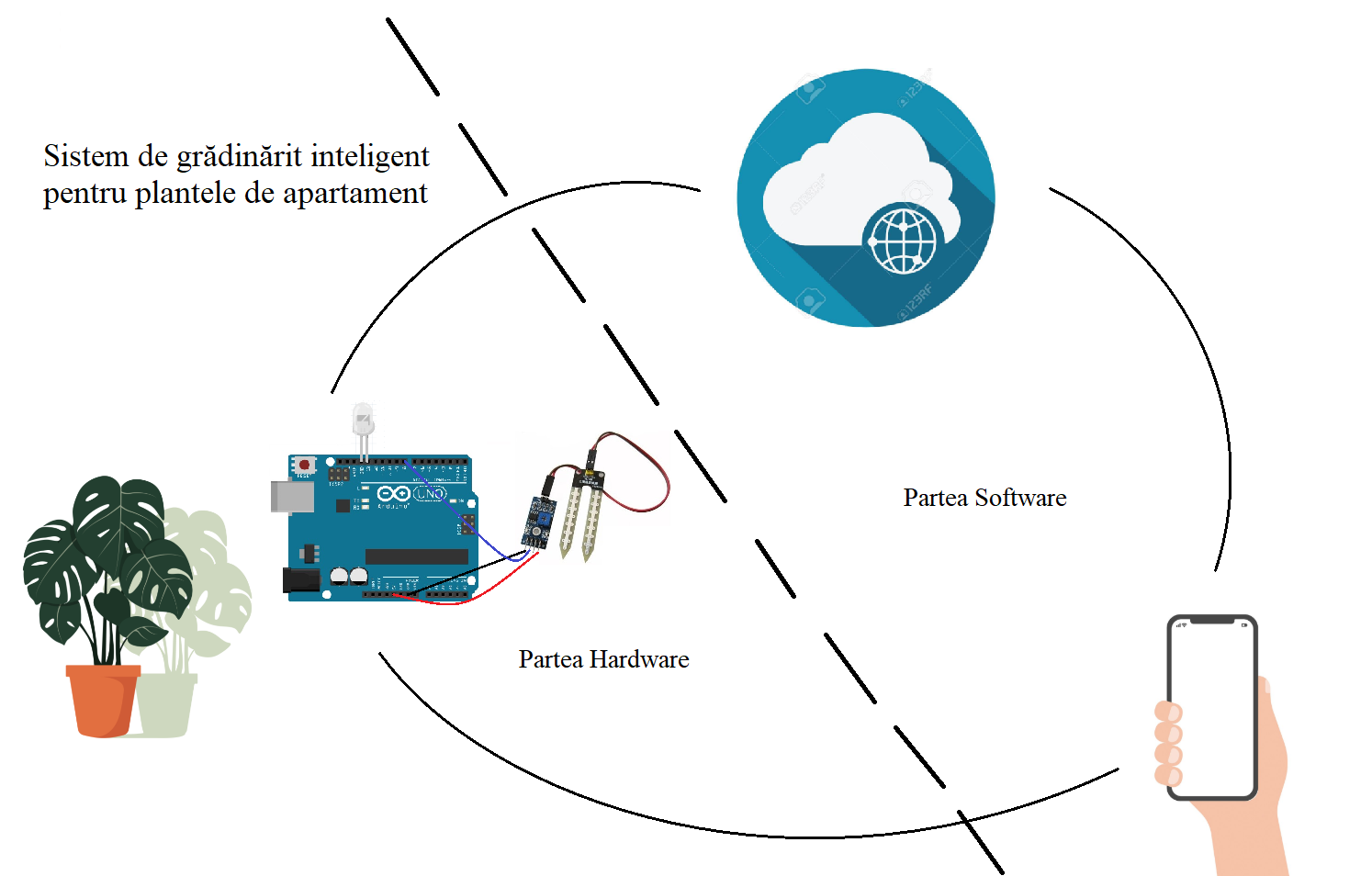


Figura . Vera – profilul unei plante

Folosind aplicațiile prezentate ca bază de pornire, lucrarea de față își propune să creeze o aplicație care să ofere componenta de monitorizare și gestionare a plantelor și a intervalului de udare sau a reglării umidității solului. Funcționalitatea de identificarea a plantei sau a nevoilor acestora poate prezenta o dezvoltare ulterioară. Pentru prototipul actual am decis să ne axăm pe monitorizarea și creșterea plantelor cunoscute.

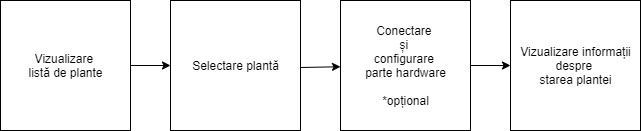
# Analiză şi Fundamentare Teoretică

Considerând obicetivele și specificațiile menționate în capitolul 2, putem împărți aplicația în două părți. Luăm această decizie deoarece putem oferi o imagine mai cuprinzătoare și mai ușor de înțeles și analizat dacă luăm în considerare aceste părți componente pe rând. În figura 4.1 se observă că partea hardware se referă la componentele care monitorizează, controlează și alterează mediul plantei iar partea software se referă la aplicația web disponibilă utilizatorilor. Când vorbim de partea software includem atât aplicația server cât și cea client.



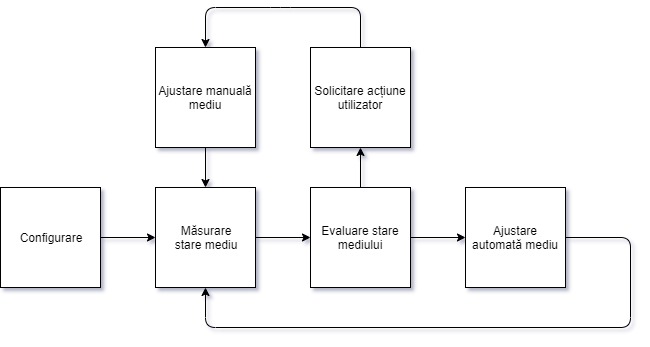
. Împărțirea aplicației

Prima parte este partea software, accesibilă gratuit oricărui utilizator. Aceasta are ca scop evidența plantelor din apartent și gestionarea programului de udare a acestora. Principalul flux al părții software este reprezentat în figura 4.2. Utilizatorul începe prin a vizualiza lista de plante disponibile în aplicație și a selecta una dintre aceste plante. Selectarea unei plante implică adăugarea acesteia în grădina virtuală a utilizatorului. Pasul următor este configurarea sistemul hardware. A se menționa că din cei patru mari pași descriși, pasul numărul trei este facultativ. Acest pas are sens doar pentru utilizatorii care beneficiază și de parte hardware a aplicației. Ultimul pas este reprezentat de vizualizare datelor desprea plantă. Acestea pot fi date și despre starea mediului plantei în cazul în care utilizatorul beneficiază de parte hardware.



. Fluxul de business al părții software a aplicației

A doua parte a aplicației este parte de hardware. Aceasta are ca scop oferirea unui mediu supravegheat și parțial controlat de creștere a plantei. Fluxul de lucru al părții hardware este evidențiat în figura 4.3. Se poate observa că primul pas e cel de configurare. După configurare, parte hardware a aplicației intră într-o buclă de măsurare și evaluare (conform configurării) a mediului plantei. În cazul în care e nevoie de ajustare automată, sistemul va ajusta singur condițiile de mediu. În cazul în care este nevoie de ajutare manuală, sistemul va notifica utilizatorul cu privire la ce acțiuni trebuie acesta să întreprindă pentru a oferi plantei un mediu propice de dezvoltare.



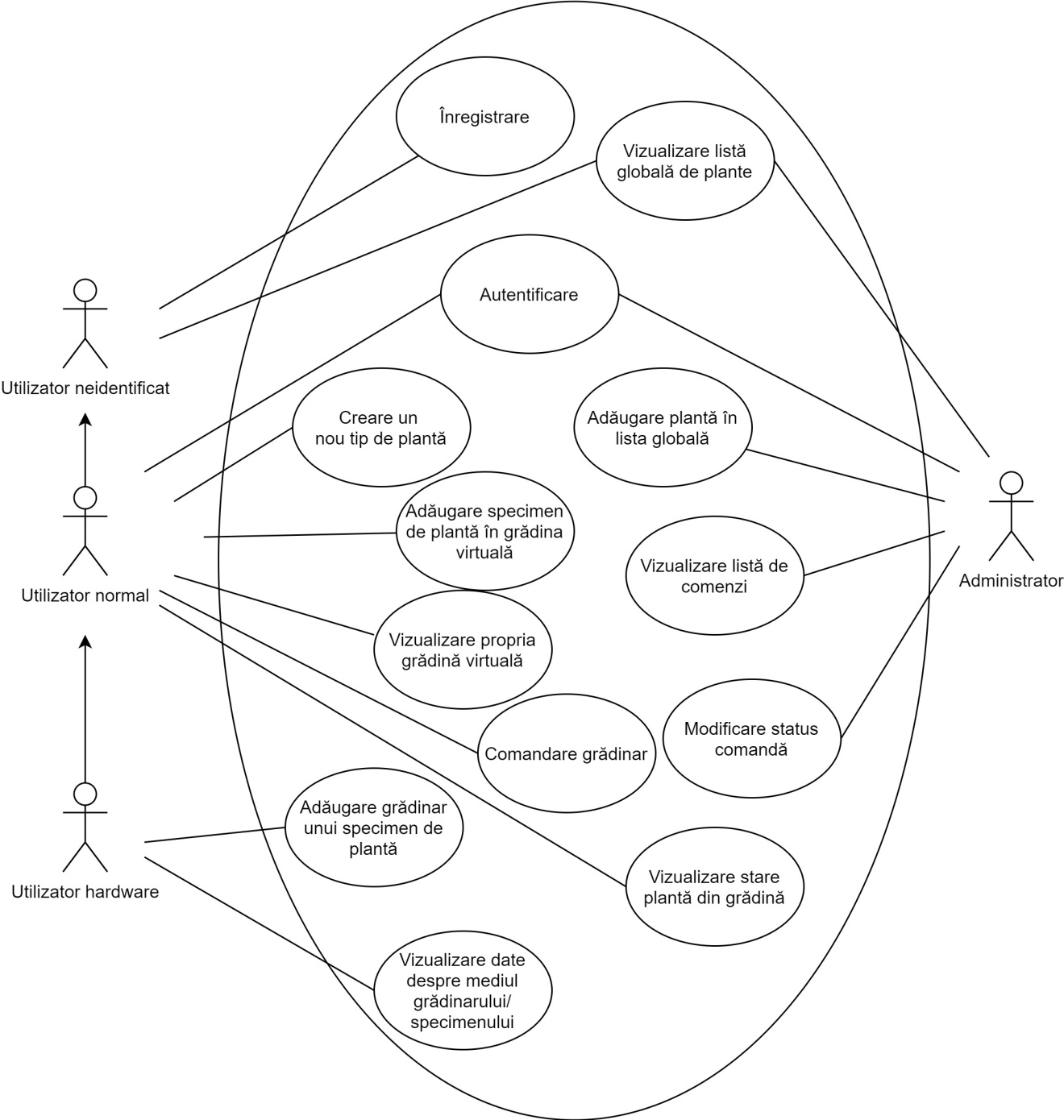
. Fluxul de lucru al părții hardware a aplicației

O privire mai în detaliu asupra aplicației, pe lângă fluxurile principale ale acesteia o oferă cazurile de utilizare ale aplicației. Spre deosebire de cerințele funcționale absolut necesare prezentate în capitolul 2, în urma unei analize mai în detaliu am decoperit nevoia de anumite cerințe funcționale adjuvante pentru aplicația noastră.

În subcapitolul următor vom prezenta în detaliu cazurile de utilizare pentru fiecare cerință funcțională.

## Cazuri de utilizare

În capitolul 2 am prezentat cerințele funcționale ale aplicației. Acestea le-am analizat mai în detaliu și au rezultat cazurile de utilizare ale aplicației. Ele sunt prezentate în figura 4.4.



. Cazuri de utilizare a aplicație

Observăm în figura 4.4 cazurile de utilizare ce vor fi detaliate ulterior și patru roluri:

* Utilizator neautentificat – rolul este reprezentativ pentru vizitorii aplicație software care nu dețin un cont
* Utilizator normal – rolul e reprezentativ pentru persoana care are un cont în aplicație
* Utilizator hardware – rolul este reprezentativ pentru persoana care are un cont în aplicație și deține și partea hardware a sistemului. Acest rol extinde roulul de utilizator normal
* Administrator – rolul e reprezentat de persoanele care administrază lista globală de plante a aplicației.

### Înregistrare

Cazul de utilizare este prezentat în tabelul 4.1. De menționat că aplicația poate fi utilizată și fără un cont, dar crearea acestuia oferă mai multe funcționalități utilizatorului.

Tabel . Caz de utilizare – Înregistrarea unui utilizator în aplicație

|  |  |
| --- | --- |
| **Descrierea cazului de utilizare** | Utilizatorul își face cont în cadrul aplicației pentru a beneficia de mai multe funcționalități. |
| **Actori** | Utilizator normal |
| **Precondiții** | Acces la internet  Acces la aplicație |
| **Postcondiții** | Un nou cont a fost creat în aplicație.  Utilizatorul are acces la aplicație cu numele de utillizator specificat în formular.  Utilizatorul are acces la aplicație cu parola specificată în formular.  Administratorul poate vedea noul utilizator în lista de utilizatori. |
| **Flux principal** | 1. Utilizatorul accesează pagina de înregistrare 2. Utilizatorul își introduce adresa de e-mail 3. Utilizatorul își introduce parola 4. Utilizatorul confirmă parola 5. Utilizatorul apasă butonul „Înregistrare” |
| **Extensii** | 2a\_1. Utilizatorul nu a introdus o adresă de e-mail validă.  2a\_2. Utilizatorul este informat de către sistem că adresa de e-mail trebuie să fie într-un format valid  3a\_1. Utilizatorul introduce o parolă mai scurtă de 6 caractere.  3a\_2. Utilizatorul este informat de către sistem că trebuie să folosească o parolă de cel puțin 6 caractere  4a\_1. Utilizatorul nu a introdus aceeași parolă în câmpul de confirmare a parolei  4a\_2. Utilizatorul este informat de către sistem că trebuie să introducă același text în câmpul de introducere a parolei și în cel de confirmare a acesteia  5a\_1. Un utilizator este deja înregistrat în aplicație cu această adresă de e-mail.  5a\_2. Utilizatorul este informat de către sistem să verifice adresa de e-mail folosită iar în cazul în care aceasta este corectă să ceară resetarea parolei. |

### Autentificare

Cazul de utilizare pentru autentificarea utilizatorilor în aplicație este prezentat în tabelul 4.2. De menționat că doar utilizatorii normali au nevoie să se înregistreze în prealabil. În cazul administratorilor, conturile acestora sunt create manual în sistem.

Tabel . Caz de utilizare – Autentificarea utilizatorilor în aplicație

|  |  |
| --- | --- |
| **Descrierea cazului de utilizare** | Utilizatorul se autentifică în aplicație pentru a beneficia de mai multe funcționalități. |
| **Actori** | Utilizatorul normal  Administratorul |
| **Precondiții** | Acces la internet  Acces la aplicație  Utilizatorul normal și-a creat cont în aplicație |
| **Postcondiții** | Utilizatorul este autentificat în aplicație.  Utilizatorul este redirecționat spre o pagină specifică. |
| **Flux principal** | 1. Utilizatorul accesează pagina de autentificare 2. Utilizatorul introduce adresa de e-mail în câmpul corespunzător 3. Utilizatorul introduce parola în câmpul corespunzător 4. Utilizatorul apasă butonul de autentificare |
| **Extensii** | 2a\_1. Utilizatorul nu a introdus o adresă de e-mail validă.  2a\_2. Utilizatorul este informat de către sistem că adresa de e-mail trebuie să fie într-un format valid  4a\_1. Utilizatorul a introdus o adresă de e-mail greșită.  4a\_2. Utilizatorul este informat de către sistem că această adresă de e-mail nu figurează cu un cont în aplicație  4b\_1. Utilizatorul a introdus o parolă greșită.  4b\_2. Utilizatorul este informat de către sistem că a introdus o parolă greșită și este invitat să corecteze parola sau să recurgă la funcționalitatea de resetarea a parolei |

### Vizualizare listă globală de plante

Cazul de utilizare este descris în tabelul 4.4. Lista global disponibilă de plante despre care aplicația deține informații. Nu este o listă exhaustivă de plante. De menționat că acesta este singurul caz pentru care nu este nevoie de autentificare.

Tabel . Caz de utilizare – Vizualizare listă globală de plante

|  |  |
| --- | --- |
| **Descrierea cazului de utilizare** | Vizualizarea listei de plante din aplicație pentru identificarea nevoilor unei plante |
| **Actori** | Utilizator normal  Utilizator neautentificat |
| **Precondiții** | Acces la internet  Acces la aplicație |
| **Postcondiții** | Acces la pagina cu lista globală de plante |
| **Flux principal** | 1. Utilizatorul accesează aplicația 2. Utilizatorul accesează lista globală de plante |
| **Extensii** |  |

### Adăugare plantă în lista globală

Având în vedere statutul de prototip al aplicației, în fază incipientă, lista globală de plante nu va fi numeroasă. Pentru ca aceasta să poată fi extinsă în timp există funcționalitatea de adăugare de noi plante de către un administrator. Cazul de utilizare este descris în tabelul 4.5.

Tabel . Caz de utilizare – Adăugare plantă în lista globală

|  |  |
| --- | --- |
| **Descrierea cazului de utilizare** | Adăugare plantă în lista globală de plante pentru extinderea acesteia |
| **Actori** | Administrator |
| **Precondiții** | Acces la internet  Acces la aplicați  Actorul este autentificat în aplicație |
| **Postcondiții** | O nouă plantă este adăugată în lista globală  Planta adăugată este vizibilă în lista globală fără autentificare |
| **Flux principal** | 1. Administratorul accesează lista globală de plante 2. Administratorul apasă butonul de adăugare o nouă plantă 3. Administratorul este redirecționat spre pagina de completare a formularului cu date despre plantă 4. Admnistratorul completează câmpurile de date cu informații relevante despre planta în cauză 5. Administratorul salvează noua plantă 6. Administratorul este redirecționat spre lista globală de plante |
| **Extensii** | 4a\_1. Administratorul nu a completat un câmp de introducere a datelor  4a\_2. Administratorul este informat de către aplicație că toate câmpurile sunt obligatorii. |

### Vizualizare grădina virtuală

Deoarece scopul aplicației este de a ajuta utilizatorul să își crească plantele personale, o imagine de ansamblu asupra acestora în cadrul aplicației este absolut necesară. În grădina virtuală utilizatorul poate vedea toate plantele pe care acesta le-a adăguat pe parcursul timpului. Ea este disponibilă doar pentru utilizatorul în cauză, niciun alt utilizator nu este capabil să vadă altă grădina virtuală decât a sa.

Tabel . Caz de utilizare – Vizualizare grădina virtuală

|  |  |
| --- | --- |
| **Descrierea cazului de utilizare** | Utilizatorul dorește să vizulizeze grădina sa virtuală |
| **Actori** | Utilizator normal |
| **Precondiții** | Acces la internet  Acces la aplicație  Utilizatorul este autentificat în aplicație |
| **Postcondiții** | Grădina virtuală este afișată |
| **Flux principal** | 1. Utilizatorul accesează aplicația 2. Utilizatorul este redirecționat spre pagina grădinii virtuale |
| **Extensii** |  |

### Adăugare plantă în grădina virtuală

Funcționalitatea de adăugare plantă în grădina virtuală deservește nevoia utilizatorului de a-și crea grădina virtuală cu o schemă a plantelor reprezentativă situației din viața reală. Cazul de adăugare a unei plante în grădina virtuală este prezentat în tabelul 4.7.

Tabel . Caz de utilizare – Adăugare plantă în grădina virtuală

|  |  |
| --- | --- |
| **Descrierea cazului de utilizare** | Utilizatorul dorește să își adauge o plantă în grădina virtuală din cadrul aplicației. |
| **Actori** | Utilizator normal |
| **Precondiții** | Acces la internet  Acces la aplicație  Utilizatorul este autentificat în aplicație |
| **Postcondiții** | Planta a fost adăugată în aplicație în cadrul grădinii virtuale a utilizatorului.  Planta este de tipul selectat (sau configurat) de către utilizator.  Planta este vizibilă în grădina virtuală a utilizatorului.  Planta nu este vizibilă pentru oricare alt utilizator. |
| **Flux principal** | 1. Utilizatorul accesează grădina virtuală din aplicație 2. Utilizatorul apasă butonul de adăugare o nouă plantă 3. Utilizatorul este redirecționat spre pagina de adăugare plantă 4. Utilizatorul introduce date specifice care să identifice exemplarul în grădina sa virtuală. 5. Utilizatorul selectează opțiunea că știe ce tip de plantă urmează să adauge 6. Aplicația afișează lista de tiputi de plante existente (din lista globală de plante) 7. Utilizatorul caută prin lista de tipuri de plante existente 8. Utilizatorul identifică tipul de plantă pe care vrea să o adauge 9. Utilizatorul selectează tipul de plantă 10. Utilizatorul apasă butonul de adăugare în grădina virtuală 11. Utilizatorul este redirecționat spre grădina sa virtuală |
| **Extensii** | 8a\_1. Utilizatorul nu identifică în lista globală de plante niciun tip de plantă care se pretează pentru expemplarul său  8a\_2. Utilizatorul selectează opțiunea că vrea să creeze un nou tip de plantă configurabil manual.  8a\_3. Utilizatorul configureză un nou tip de plantă pentru exemplarul său  8a\_4. Noul tip de plantă este disponibil în lista cu tipuri de plantă existente (doar pentru utilizatorul în cauză) |

### Creare tip de plantă manual

Întrucât aplicația nu deține o listă exhaustivă de plante, unii utilizatori e posibil să nu găsească în lista globală de plante un tip de plantă adecvat exemplarului pe care aceștia îl dețin. În acest caz, oferim utilizatorilor posibilitatea ca aceștia să creeze un nou tip de plantă în cadrul grădinii lor virtuale pe care sa îl poată utiliza pentru adăugarea de noi exemplare.

Tabel . Caz de utilizare – Creare plantă configurată manual

|  |  |
| --- | --- |
| **Descrierea cazului de utilizare** | Creare tip de plantă configurabil manual |
| **Actori** | Utilizator normal |
| **Precondiții** | Acces la internet  Acces la aplicație  Utilizatorul este autentificat  Utilizatorul este în cazul de utilizare „Adăugare plantă în grădina virtuală” și a executat pasul 5 |
| **Postcondiții** | Un nou tip de plantă a fost creat  Tipul de plantă este disponibil utilizatorului care l-a creat  Tipul de plantă nu este disponibil altor utilizatori |
| **Flux principal** | 1. Utilizatorul selectează opțiunea că știe ce tip de plantă urmează să adauge 2. Utilizatorul selectează opțiunea de creare a unui nou tip de plantă confgurabilă manual 3. Aplicația afișează secțiunea formularului de creare tip de plantă configurabil manual 4. Utilizatorul completează câmpurile de date din formular. 5. Utilizatorul salvează noul tip de plantă |
| **Extensii** | 4a\_1. Utilizatorul nu a completat un câmp de date obligatoriu  4a\_2. Utilizatorul este informat de către aplicație că toate câmpurile de date sunt obligatorii |

### Vizualizare infomații plantă din grădină

Vizualizarea informațiilor unei plante din grădina virtuală se adresează și utilizatorilor care nu au acces la partea hardware a aplicației. Aici se regăsesc informații despre necesitătile acestui exemplar în funcție de tipul plantei și infomațiile specifice pe care utilizatorul le-a salvat despre aceasta. Informații specifice se referă la dimensiunea plantei, anotimpul curent, luminozitatea de care aceasta beneficiază. Cazul de utilizare este prezentat în tabelul 4.10.

Tabel . Caz de utilizare – Vizualizare informații plantă din grădină

|  |  |
| --- | --- |
| **Descrierea cazului de utilizare** | Utilizatorul dorește să vizualizeze informații despre un exemplar din grădina sa virtuală |
| **Actori** | Utilizator normal |
| **Precondiții** | Acces la internet  Acces la aplicație  Utilizatorul este autentificat în aplicație  Utilizatorul deține cel puțin un exemplar în grădina sa virtuală |
| **Postcondiții** | Aplicația afișează informații despre exemplarul pe care l-a selectat utilizatorul |
| **Flux principal** | 1. Utilizatorul navighează la pagina grădinii virtuale. 2. Utilizatorul apasă pe unul dintre exemplarele din grădina virtuală 3. Aplicația afișează informații relevante cu privire la exemplarul selectat de utilizator |
| **Extensii** |  |

În figura 4.5 am evidențiat printr-o diagramă de secvență cazurile de utilizare pentru înregistrare, autentificare, vizualizare listă de plante, adăugare tip de plantă, adăugare specimen de plantă și vizualizarea nevoilor acestuia în grădina utilizatorului.

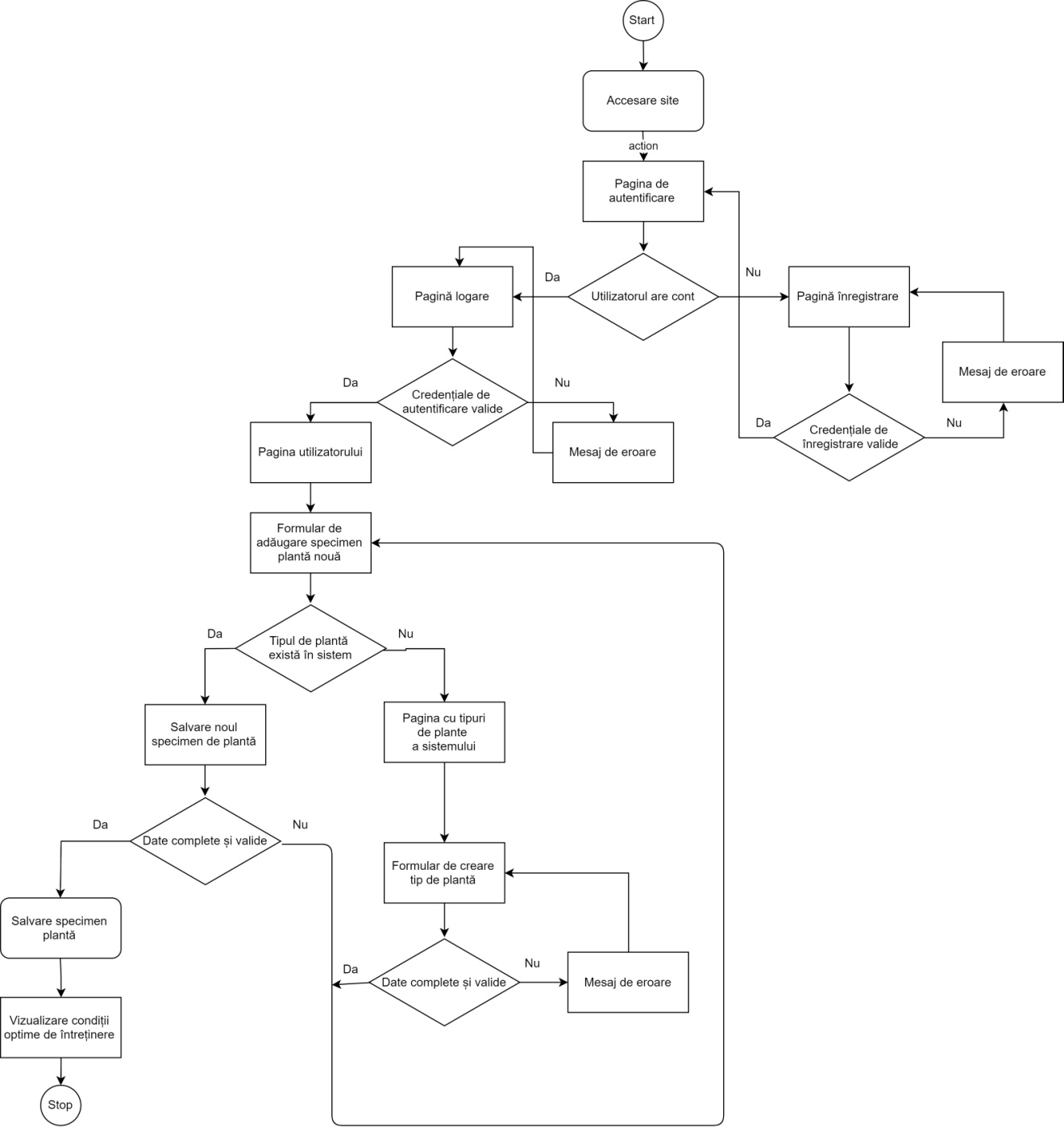


Figura . Diagramă de secvență pentru crearea unui specimen nou (include toate cazurile de utilizare care sunt postcondiții pentru acesta)

### Comadare grădinar - partea hardware

Această funcționalitate se adresează doar utilizatorilor care doresc să aibe acces și la partea hardware a aplicației. De menționat că partea harware furnizată unui utilizator este configurată din fabrică special pentru contul utilizatorului respectiv. Un utilizator poate avea mai multe părți hardware care aparțin aceluiași cont (pentru mai multe plante din apartament) dar o componentă hardware nu poate monitoriza două plante și nici nu poate fi accesată din două conturi. Cazul de utilizare pentru comandarea unui grădinar de ghiveci este prezentat în tabelul 4.11.

Tabel . Caz de utilizare – Configurare parte hardware

|  |  |
| --- | --- |
| **Descrierea cazului de utilizare** | Utilizatorul dorește să configureze partea hardware a aplicației după nevoile plantei |
| **Actori** | Utilizator hardware  Administrator |
| **Precondiții** | Acces la internet  Acces la aplicație  Utilizatorul este autentificat în aplicație |
| **Postcondiții** | Partea hardware a sistemului este configurată conform specificațiilor și livrată utilizatorului.  Parte hardware este reprezentată în sistem sub forma unei entități de tipul „Grădinar” |
| **Flux principal** | 1. Utilizatorul accesează pagina „Grădina mea” 2. Utilizatorul apasă butonul de comandare grădinar. 3. Utilizatorul introduce datele necesare. 4. Sistemul crează o comandă în statusul „PENDING” 5. Un administrator acceptă comanda 6. Comanda ajunge în statusul „ACCEPTED” 7. Administratorul și utilizatorul au o discuție live în care stabiliesc specificațiile ghiveciul cu care vine grădinarul semiautomat 8. Administratorul trimte prin curier ghiveciul cu grădinar utilizatorului 9. Administratorul actualizează comanda în statusul „COMPLETED” 10. Sistemul crează o reprezentarea a grădinarului |
| **Extensii** | 3a\_1. Datele introduse de utilizator nu sunt complete.  3a\_2. Sistemul afișează un mesaj de eroare.  3a\_3. Utilizatorul corectează datele |

### Vizualizare date despre mediul plantei

Acest scenariu se adresează doar utilizatorilor care au și dispozitivul hardware. Cazul de utilizare este prezentat în tabelul 4.12.

Tabel . Caz de utilizare – Vizualizare date despre mediul plantei

|  |  |
| --- | --- |
| **Descrierea cazului de utilizare** | Utilizatorul doreșre să vadă informații despre mediul în care se află planta și dacă acestea sunt în parametri normali pentru dezvoltarea optimă a plantei. |
| **Actori** | Utilizator hardware |
| **Precondiții** | Acces la internet  Acces la aplicație  Utilizatorul deține modulul hardware al aplicației  Modulul hardware al aplicație este configurat |
| **Postcondiții** | Afișarea stării mediului și a analizei acesteia |
| **Flux principal** | 1. Utilizatorul accesează pagina contului său 2. Utilizatorul vizualizează date înregistrate de grădinarul semiautomat 3. Aplicația afișează utilizatorului ultimele valori înregistrate de către modulul hardare și cât de benefice sunt considerate acestea pentru exemplarul configurat. |
| **Extensii** |  |

În figura 4.6 am evidențiat printr-o diagramă de acțiuni cazurile de utilizare 4.1.9 și 4.1.10 pentru comandarea unui grădinar semiautomat și vizualizarea datelor înregistrate de acesta.

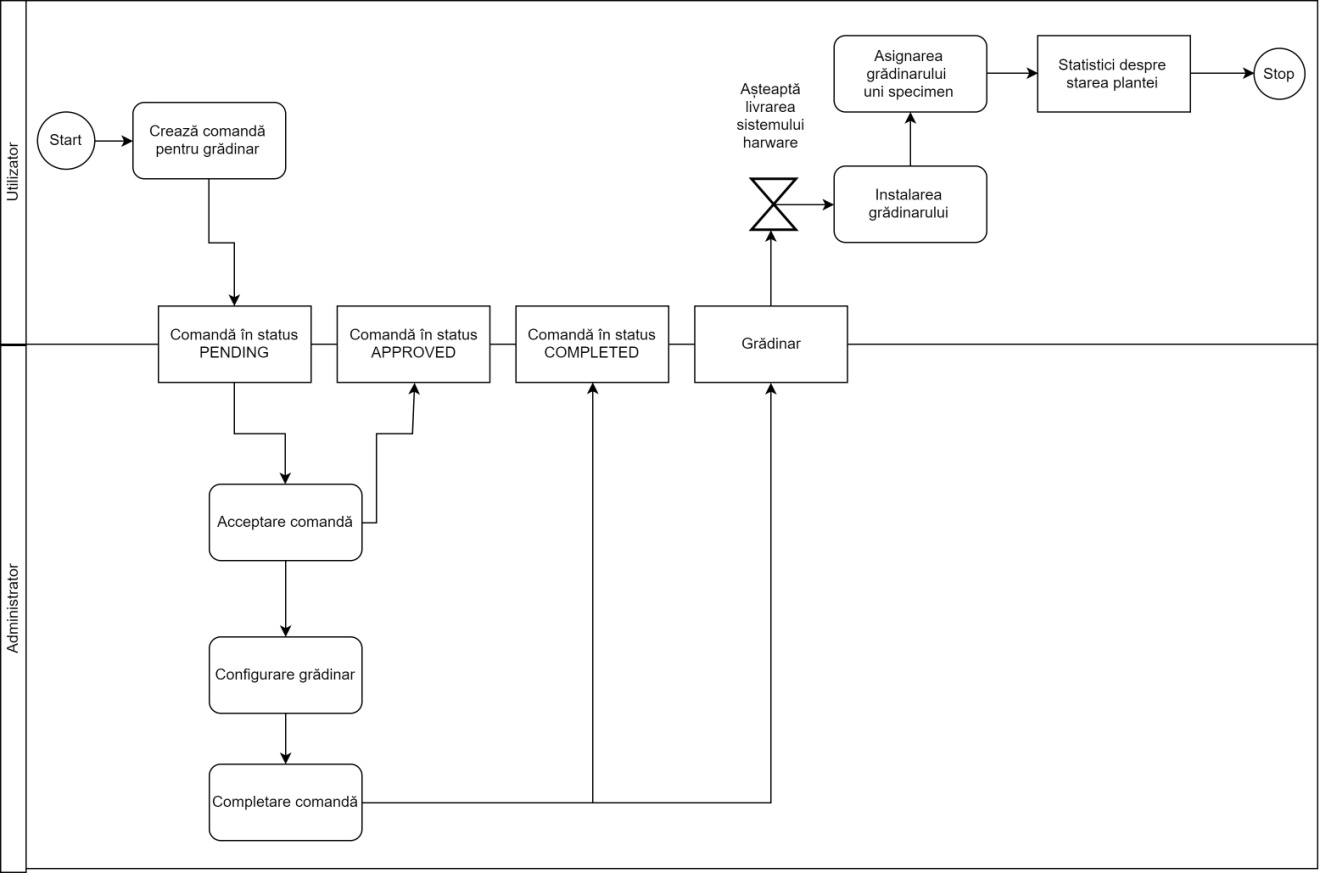


Figura . Comandare și livrarea unui grădinar semiautomat

## Module

După prezentarea cazurilor de utilizare ale aplicației și o scurtă analiză a acestora, aplicația poate fi împărțită în patru module principale:

* Modulul de gestionarea a utilizatorilor
* Modulul de gestionare a plantelor
* Modulul de grădinărit semi-automat

Am ales această împărțire deoarece izolează nu doar tipul obiectelor pe care fiecare modul le gestionează, dar și motivele pentru care fiecare modul s-ar putea schimba. Pentru a respecta principul responsabilității unice („Single Responsibility Principle”) [27] luăm în considerare motivele pentru care poate fi nevoie ca un modul să își schimbe funcționalitățile. Astfel, dacă vom dori o nouă formă de autentificare sau un nou tip de utilizator, va trebui să modificăm logica dintr-un singur modul.

În continuare vom prezenta responsabilitățile fiecărui modul în parte și interacțiunile de care e nevoie între acesta și celelalte module pentru implementarea cazurilor de utilizare specificate anterior.

### Modulul de gestionare a utilizatorilor

După cum îi spune și numele acest modul se ocupă cu gestionarea utilizatorilor aplicației. În cazul de față gestionare înseamnă validare, stocare, accesare și procesare de date despre utilizatori. Principalele responsabilități ale acestui modul sunt:

* Înregistrarea unui utilizator nou
* Autentificarea unui utilizator existent
* Verificarea permisiunilor de acces a unui utilizartor existent asupra unei anumite resurse

Acest modul are o interfață directă către utilizator, deci va avea nevoie de pagini și formulare web sugestive, care oferă feedback rapid și sunt ușor de utlizat. De exemplu pentru validarea datelor, în cazurile posibile, pentru un feedback rapid către utilizator, validarea ar trebui făcută instantaneu în front-end.

Acest modul este de sine stătător și poate funcționa fără interacțiunea cu alte module. De menționat că pentru implementarea funcționalităților discutate, alte module au nevoie de a interacționa cu modulu de gestionare a utilizatorilor.

### Modulul de gestionare a plantelor

Acest modul se ocupă de două mari domenii: cel de gestionarea a tipurilor de plante și cel de gestionare a plantelor ca exemplare. Motivul pentru care am păstrat ambele domenii într-un singur modul este că nu am întrevăzut un tip de schimbare asupra funcționalităților existente în cadrul tipurilor de plante care să nu aducă după ea nevoia unei modificari în cadrul exemplarelor de plante și vice versa.

Modulul de gestionarea a plantelor este format din functionalitățile de

* Vizualizare a tipurilor de plante existente
* Adăugarea unui nou tip de plantă
* Validarea deplinătății infomațiilor tipului de plantă
* Vizualizarea exemplarelor de plante din grădina unui utilizator
* Adăugarea unui nou exemplar în grădină
* Vizualizarea informațiilor despre un exemplar din grădină (a nu se confunda cu vizualizarea datelor colectate de echipamentul hardware)

Și acest modul are o interfață directă către utilizator, deci va avea nevoie de pagini și formulare web sugestive, care oferă feedback rapid și sunt ușor de utlizat. De asemenea, acest modul va avea nevoie de un design atratctiv, acesta fiind modulul cu care vor interacționa cel mai des majoritatea utilizatorilor.

Acest modul interacționează cu (și depinde de) modulul de gestionare a utilizatorilor pentru:

* Identificarea utlizatorului curent în cazul afișării informațiilor despre grădina acestuia sau a exemplarelor din ea
* Identificarea rolului utilizatorului curent în cazul adăugării de noi tipuri de plante in lista globală sau confirmarea comenzilor

### Modulul de grădinărit semi-automat

Modulul de grădinărit semi-automat reprezintă partea hardware principală a sistemului. El este responsabil pentru

* monitorizarea mediului în care este planta
* transmiterea valorilor recepționate
* recepționarea valorilor de creștere optimă
* acționarea unor dispozitive capabile să altereze starea mediului

Pentru monitorizarea mediului am ales senzori care să măsoare elementele esențiale creșterii adecvate a unei plante. Cele mai importante care intră în această categorie sunt cantitatea de apa de care dispune planta și lumina la care este expusă. Acestea sunt urmate de temperatura ambientală. (Deși temperatura este de asemenea un factor primordial în cardul dezvoltării unei plante, având în vedere că proiectul se adresează plantelor de apartament, unde temperaturile sunt în general adecvate plantelor, am considerat că în cazul nostru, temperatura este un element secundar) După apă, lumină, temperatură, am decis să integrăm și umiditatea aerului.

În figura 4.5 se poate observa legătura dintre părțile modululi. Aparatele care măsoară date despre mediu trimit informația la logica de control, aceasta, verificând setările pentru specimenul de plantă, compară valorile înregistrate cu cele configurate pentru creșterea propice, iar dacă valorile înregistrate nu sunt în parametrii adecvați, logica de control e responsabilă să acționeze sistemul de control, ca mai apoi acesta să schimbe starea mediului.

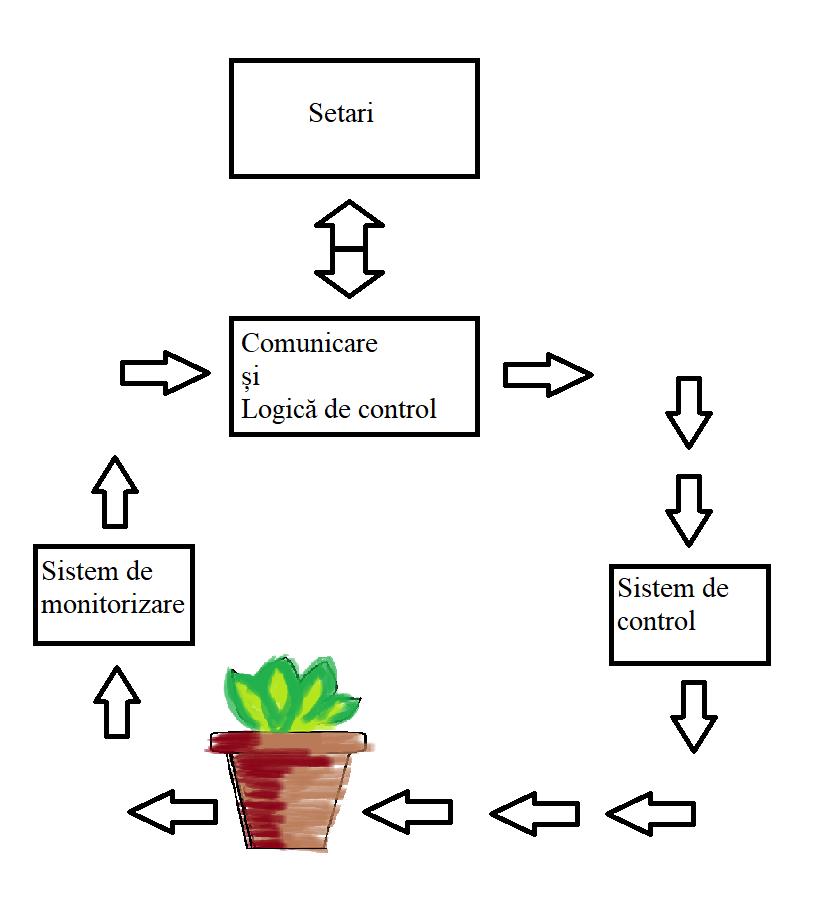


Figura . Modulul de grădinărit semi-automat

Având în vedere măsurătorile pe care le luăm din mediu, putem alege care dintre caracteristicile mediului ar trebui să fie alterabile de către sistemul semi-automat și care de către utilizator. Vom observa fiecare posibilitate în parte.

Cantitatea de apă de care dispune o plantă reprezintă, în majoritatea cazurilor, motivul pentru care plantele de apartament își încheie viață. Udatul plantelor reprezintă cea mai costisitoarea activitate din punct de vedere al timpului în întreținerea unei plante. În majoritatea cazurilor este singura activitate care trebuie făcută pe o perioadă de șase luni până la un an.

Deoarece deseori proprietarii uită să își ude plantele, automatizarea acestui element constituie un avantaj substanțial pentru sistemul de grădinărit.

Lumina de care dispune planta este și ea crucială unei dezvoltări propice. În cazul plantelor florale, este nevoie de cât mai multă lumină, deoarece acestea fără o cantitae suficientă de lumină, nu ating inflorescența. Nu este de neglijat nici cazul în care plantele au prea multă lumină, în special lumină directă. De exemplu, planta de aloe vera se arde (arsuri solare) într-o lumină prea puternică, aceasta trebuind să fie expusă doar unei lumini difuze. Pentru a putea controla cantitatea de lumină a unei plante în mod automat putem să utilizăm două sisteme diferite de control: unul care intensifică lumina în cazul în care aceasta nu este suficientă și unul care diminuează lumina în cazul în care aceasta este prea abundentă. În acest scop putem utiliza o sursă de lumină UV și un paravan cu un grad de opacitate redus.

Având în vedere că scopul plantelor de apartament este unul decorativ, și că utilizarea paravanelor său al sursei de lumină ar altera imaginea plantei, am ales să nu automatizăm acest proces. Un alt motiv al acestei alegeri este că pentru a altera condițiile de luminozitate a unei plante tot ce trebuie să facem e să îi schimbăm amplasamentul (să o punem într-o zonă mai luminoasă sau mai întunecată). Aceasta este o acțiune ce trebuie făcută o singură dată, spre deosebire de udarea plantei, astfel că nu o considerăm o acțiune la fel de incomodă pentru utilizator.

Valoarea inadecvată a temperaturii ambientale a plantelor este și aceasta una dintre cauzele morții lor premature. Cel mai întâlnit exemplu este când proprietarii își depozitează plantele iarna deasupra sau în apropierea surselor de căldură precum radiatoarele sau sobele. Pentru a putea controla temperatura ambientală a plantei, este necesară încastrarea plantei într-un container și implementarea unui sistem capabil atât să încălzească containerul respectiv (când temperatura ambientală e prea mică), cât și să îl răcească (când temperatura ambientală este prea mare). Deoarece această funcționalitate este costisitoare atât din punct de vedere estetic dar mai important energetic, și pentru că, la fel ca în cazul cantității de lumină, poate fi reglată cu ușurință de utilizator, am decis să nu o automatizăm.

Un alt aspect asupra căruia punem problema automatizării este umiditatea aerului din jurul plantei. Acest aspect nu este primordial în creșterea plantei și din această cauză, împreuna cu faptul că planta poate fi poziționată lângă aparate electrice am ales să nu îl automatizăm.

În concluzie modulul de grădinărit semiautomat va monitoriza temperatura ambientală, intensisatea luminii, umiditatea solului și a aerului dar va controla doar cantitatea de apă necesară unei plante.

# Proiectare de Detaliu si Implementare

Pentru prezentarea din punct de vedere tehnic al aplicației, spre o mai bună reprezentare, vom împărți acest capitol în mai multe secțiuni care vor acoperi, fiecare, specificații despre: aplicația server, aplicația client, modulul hardware, baza de date și legăturile între acestea. O arhitectură sumară a sistemului de grădinărit „Grădinarul de ghiveci” se poate observa în figura 5.1.

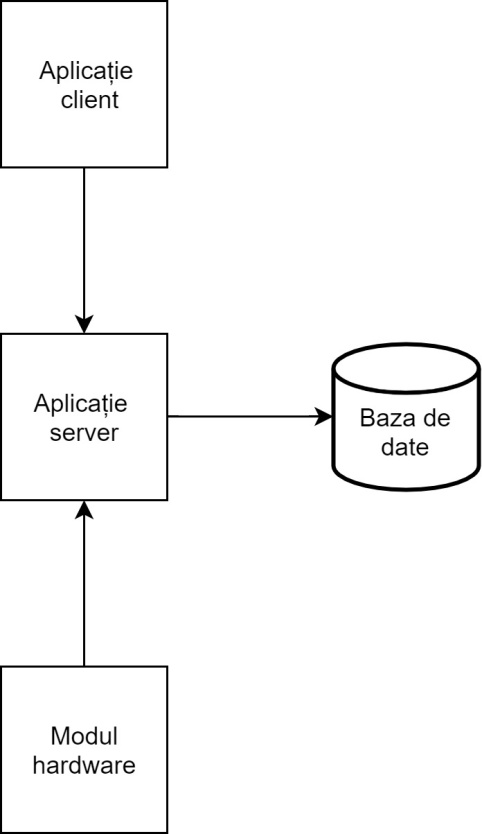


Figura . Arhitectură sumară – „Grădinarul de ghiveci”

Identificăm din figura 5.1 că aplicația client comunică doar cu aplicația server, și că nu are nicio legătură directă cu modulul hardware. Reciproca este valabilă: și modulul hardware comunică exclusiv cu aplicația server. Aplicația server răspunde mesajelor de la client și/sau modulul hardware și inițiază comunicare doar cu baza de date. Prin acest mod de segregare a componentelor am încercat să atingem o decuplare cât puternică a lor, astfel, fiecare component putând fi dezvoltat în mod cât mai independent de celelalte.

## Aplicația server

Aplicația server este centrul logic principal al sistemului. Aceasta e responsabilă de manipularea datelor. Este componenta cu care interactionează celelate module. Arhitectura acestei aplicații este „Layerd architecture”. Arhitectura Layered este formartă din:

* Prezentare – Presentation layer
* Logică – Business layer
* Persistența datelor – Persistance layer
* Baza de date – Database layer

**Presentation layer** – reprezintă interfața grafică a unui sistem. E partea cu care utilizatorul interacționează direct. În sistemul de față layer-ul prezentare este reprezentat de aplicația client așa că în aplicația server nu vom avea acest layer. Ce vom avea în schimb este o interfață prin care aplicația client să poată interacționa cu aplicația server. Acest layer este reprezentat de API (Application Programing Interface).

**Business layer** – reprezintă parte de procesare și alterare a datelor. Conform arhitecturii layered în care un layer interacționează doar cu layerele direct adiacente, acest layer va interacționa cu API-ul și cu layer-ul de persistență.

**Persistance layer** – este responsabil cu interacțiunea directă cu baza de date. Singurul celălalt layer care ar trebui să interacționeze cu acesta este cel de business.

**Database layer** – este reprezentat de baza de sistemul de stocare a datelor. Acesta poate fi o bază de date sau chiar sistemul de fișier. Pentru aplicația noastră am ales o bază de date externă aplicație server. Aceasta este prezentată în capitoulul 5.x.

În concluzie, în cazul nostru, aplicația server va consta din următoarele layere:

* API layer
* Business layer
* Persistance layer

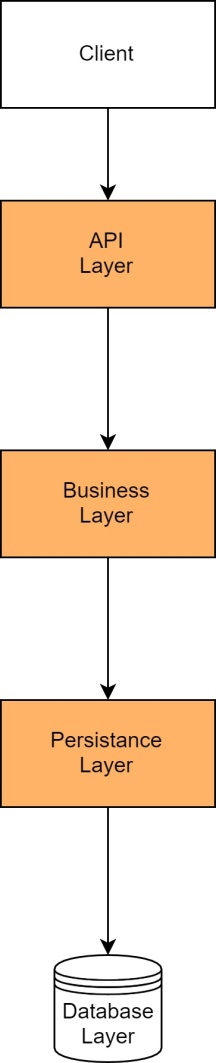


Figura . Arhitectura aplicației server

În figura 5.x observăm o arhitectura sistemul nostru în care părțile portocalii aparțin aplicației server.

Conform acestei diagrame și tiparului arhitectural layered, în aplicația noastră toate interacțiunile unui layer sunt apeluri spre layerul inferior sau răspunsurile asociate apelurilor din layerul direct superior. Am evidențiat aceasta printr-o diagramă de secvență a cazului de utilizare „Utilizatorul salvează un nou specimen” în figura 5.x. Dinou, cu portocaliu sunt componentele ce țin de aplicația server.

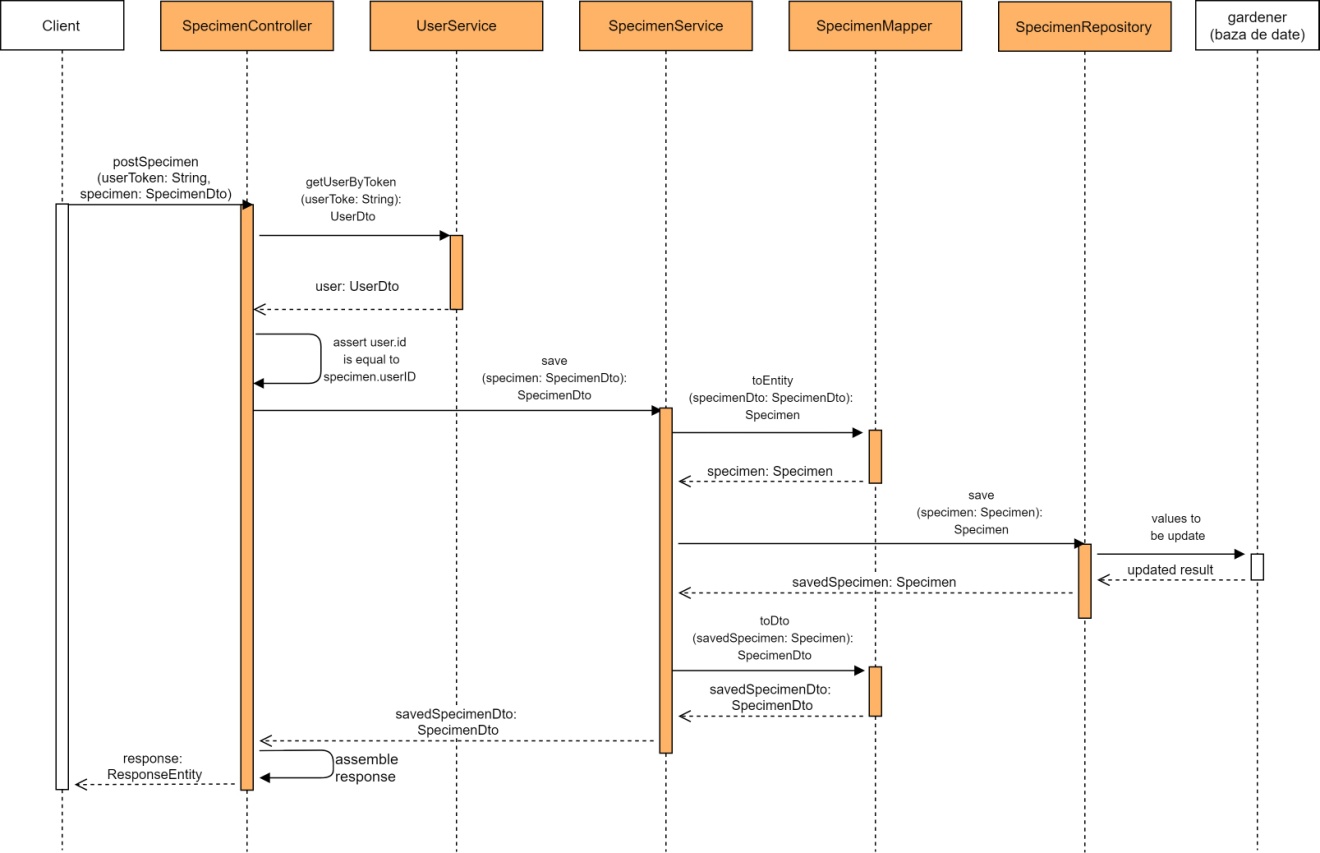


Figura . Diagramă de secvență pentru cazul de utilizare „Utilizatorul salvează un nou specimen”

Pentru implementarea acestei aplicații am folosit diverse tehnologii cu diferite scopuri. În continuare vom prezenta tehnologiile folosite, vom evidenția motivele pentru care le-am ales și vom prezenta trăsături deosebit de interesante ale acestora unde este cazul.

### Kotlin

Kotlin este un limbaj de programare bazat pe Java, pentru JVM și Android, care combină funcțiile de programare orientate pe obiecte și funcționale. Se axează pe interoperabilitate, siguranță, claritate și suport pentru „tools”.[[22]](#footnote-23) Având în vedere complexitatea medie pe care trebuie să o susțină aplicația server, și multitudinea de limbaje de programare pe care le puteam alege, există o gamă largă de opțiuni.

Alegerea Kotlin a fost motivată de popularitatea în creștere a limbajului, de curiozitate și de ușurința raportată în utilizarea lui. De asemenea de compatibilitatea cu restul tehnologiilor.

### Gradle

Gradle este un build tool care poate fi folosit pentru diverse libaje de programare. El ajută în controlul procesului de dezvoltare software prin task-uri care se ocupă cu compilarea, împachetarea, testarea, deploy-ul și publicarea aplicațiilor. Instrucțiunile Gradle se scriu într-un limbaj specific Gradle. În locul fișierleo XML, Gradle vine cu propriul DSL (Domain Specific Language) care este mai ușor de citit și scris. [[23]](#footnote-24)

Alegerea acestui tool e motivată de testarea ușurinței cu care susține că poate fi folosit. De asemenea de compatibilitatea cu restul tehnologiilor. Îl vom utiliza pentru management-ul dependențelor, compilare, împachetare, generare de cod și testare.

### Spring boot

Spring boot este un framework open-source bazat pe Java, folosit în special în dezvoltarea microserviciilor. El oferă o platformă foarte bună pentru dezvoltarea de aplicații „stand-alone”. Se poate utiliza cu un minimum de configurații. Este ușor de înțeles și de folosit deoarece renunță la utilizarea fișierleor XML pentru configurare și se bazează pe adnotările claselor.[[24]](#footnote-25)

Spring oferă și o platformă în care putem configura și genera scheletul unei aplicații de tipul Spring Boot[[25]](#footnote-26).În figura 5.x putem observa opțiunile pe care le-am selectat pentru proiectul nostru.

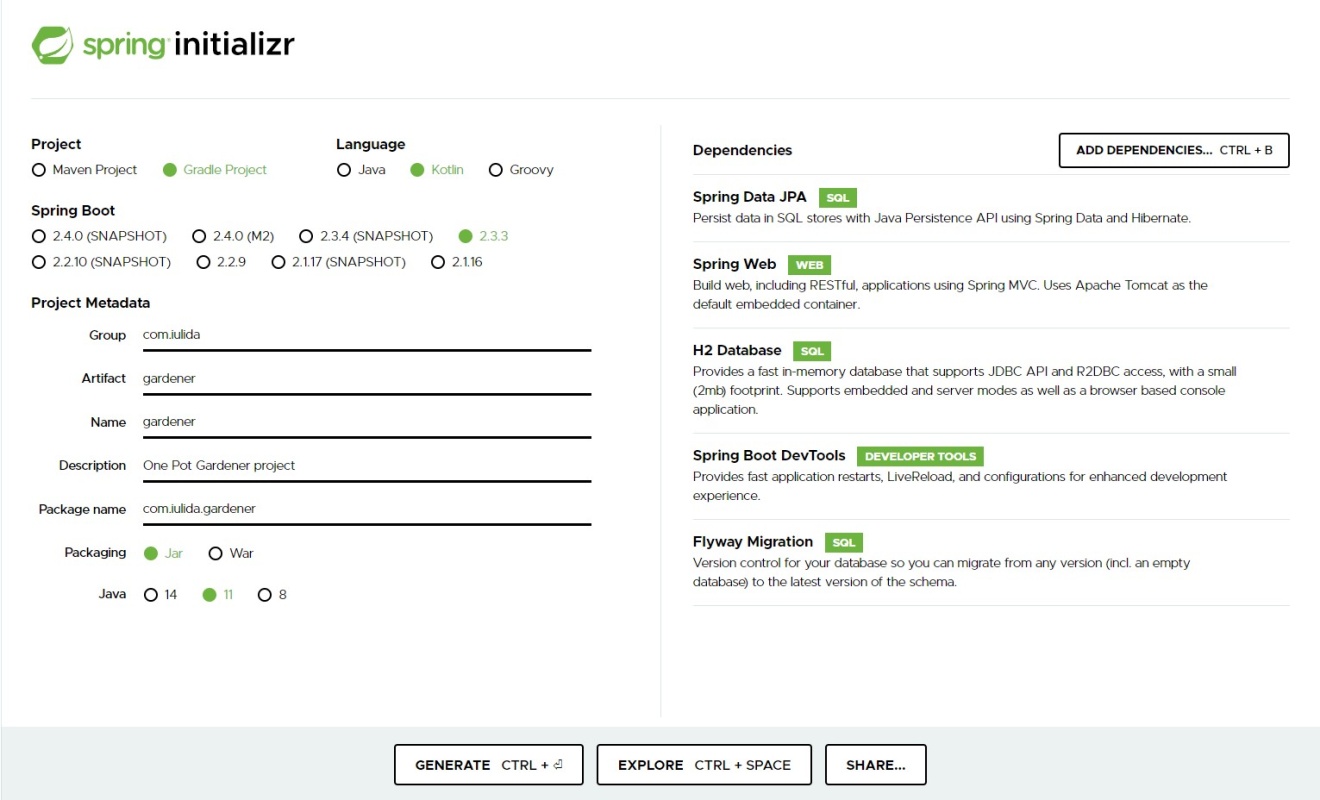


Figura . Generarea scheletului aplicației server

După cum se observă, cu spring initializr am putut crea scheletul unei proiect Gradle scris în Kotlin cu foarte mare ușurință. De asemenea, a fost posibilă adăugarea unor dependințe de bază. Deși a necesitat mai mult studiu incipient, această alegere a fost profitabilă din punct de verede al timpului de dezvoltare.

### OpenAPI

După cum am prezentat la începutul capitolului 5, aplicația server va avea în locul layer-ului de prezentare un layer de API[[26]](#footnote-27). Un API este o interfață care definește modul de interacțiune dintre două tipuri de actori: server și client. Pentru dezvoltarea API-ului în cadrul aplicației server am apelat la o soluție automată bazată pe standardul OpenAPI Specification[[27]](#footnote-28) și pe tool-ul OpenAPI Generator[[28]](#footnote-29).

OpenApi Specification este un standard pentru descrierea API-urilor REST (Representational State Transfer) care este ușor de citit și de utilizator și ușor de interpretat de calculator. El este scris în fișiere JSON[[29]](#footnote-30). În cazul de față am ales să folosim un fișier YAML[[30]](#footnote-31) (YAML Ain’t Markup Language) care este un superset al JSON dar în loc de acolade folosește indentările pentru a separa obiectele. Acest lucru îl face mai ușor de citit, motiv principal pentru care l-am ales.

În figura 5.x observăm o secțiune din fișierul de specificații OpenAPI al aplicației noastre și cum este acesta interpretat. Pentru interpretarea lui online am utilizat editorul online Swagger Editor[[31]](#footnote-32).

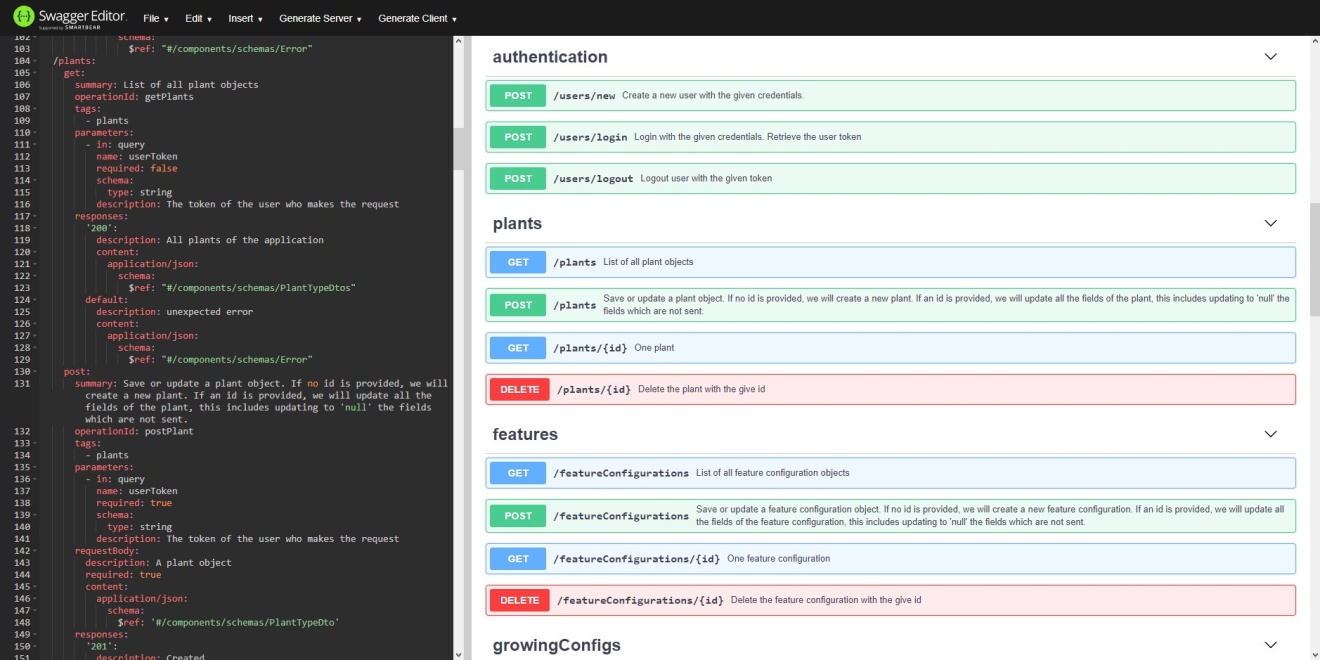


Figura . OpenAPI Specification pentru aplicația server interpretată de Editorul Swagger

Acest fișier de specificații OpenAPI poate fi utilizat pentru a genera atât documentație cum observăm în figura 5.x cât și codul reprezentativ. OpenAPI Generator deține o listă impresionantă de generatori atât pentru aplicații server cât și pentru aplicații client. În cazul nostru avem nevoie pentru aplicația server de generarea codului cu generatorul kotlin-spring[[32]](#footnote-33).

OpenAPI Generator oferă o aplicație CLI[[33]](#footnote-34) dar pentru a automatiza generarea codului API putem să ne folosim de Plugin-ul de Generare OpenAPI de Gradle[[34]](#footnote-35). Observăm în figura 5.x că după ce adăugăm acest plugin în Gradle, există niște task-uri în plus. Dintre acestea ne interesează openApiGenerate, pe care îl vom folosi pentru a genera modelele și API-ul aplicație.

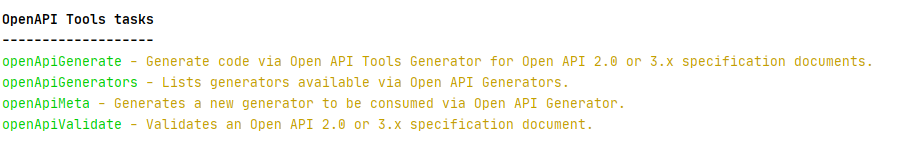


Figura . Task-uri Gradle puse la dispoziție de OpenAPI Tools

În figura 5.x observăm configurarea task-ului de generare a modelelor și serviciilor API.



Figura . Configurarea task-ului openApiGenerate pentru generarea codului de API

Un exemplu al unui endpoint generat este vizibil în figura 5.x. În parte stângă e specificarea OpenAPI iar în parte dreaptă e rezultatul generat de plugin.



Figura . Specificație OpenApi (stânga) și cod kotlin-sprin generat (dreapta)

Deși la o primă vedere e mai mult de scris în fișierul de specificație, e de reamintit că acest fișier poate fi folosit pentru generare de cod și pentru aplicații client. Aceasta este intenția noastră pentru aplicația client a sistemului de grădinărit. Aceasta este prezentată în detaliu în capitolul 5.x.

### Sendgrid

<<Sigur\_Urmează\_până\_luni\_dimineață>>

### Push API – Firebase Cloud Messaging

<<Poate\_Urmează\_până\_luni\_dimineață >>

### Spring Data JPA

Pentru implementarea layer-ului de persistență am utlizat Spring Data JPA[[35]](#footnote-36). Aceata este un modul din Spring Data care se ocupă cu suportul pentru accesarea bazei de date prin specificațiile JPA (Java Persistance API)[[36]](#footnote-37). Utilizarea lui reduce foarte mult din codul repetitiv de umplutură (boilerplate code) deoarece oferă suport pentru query-uri după valorile field-urilor cât și paginarea rezultatelor.

În figura 5.x putem observa cum utilizând Spring Data JPA un query pe baza de date devine o singură apelare de metodă.

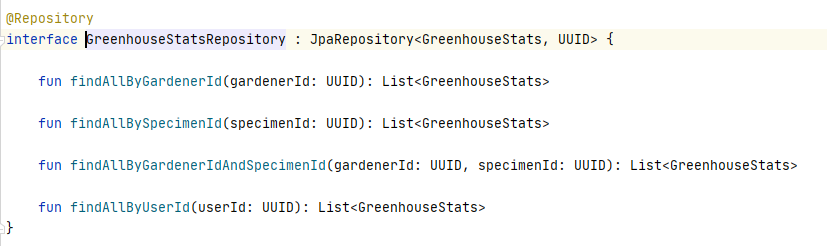


Figura . Beneficiile Spring Data JPA evidențiate prin claritatea codului

## Baza de date

### PostgreSQL

Pentru persistarea datelor despre plante, a măsurătorilor despre acestea, a datelor despre „grădinari”, a datelor desprepre utilizatori și relațiile dintre aceste entități am ales să folosim o bază de date relațională. Pentru implementarea souluției am folosit baza de date relațională open-source PostgreSQL[[37]](#footnote-38). PostgreSQL este o bază de date open-source care a început ca proiectul POSTGRES la Universitatea Berkeley din California iar acum, cu o vechime de peste 30 de ani, vin în ajutorul dezvoltatorilor cu funcționalități care stochează și scalează și cele mai complicate tipuri de date.

Pentru vizualizarea datelor am folosit aplicația DBVisualizer[[38]](#footnote-39) cu ajutorul căreia am generat o diagramă ERM (Entity Relationship Diagram) a baze de date. Aceasta se poate observa în figura 5.x.

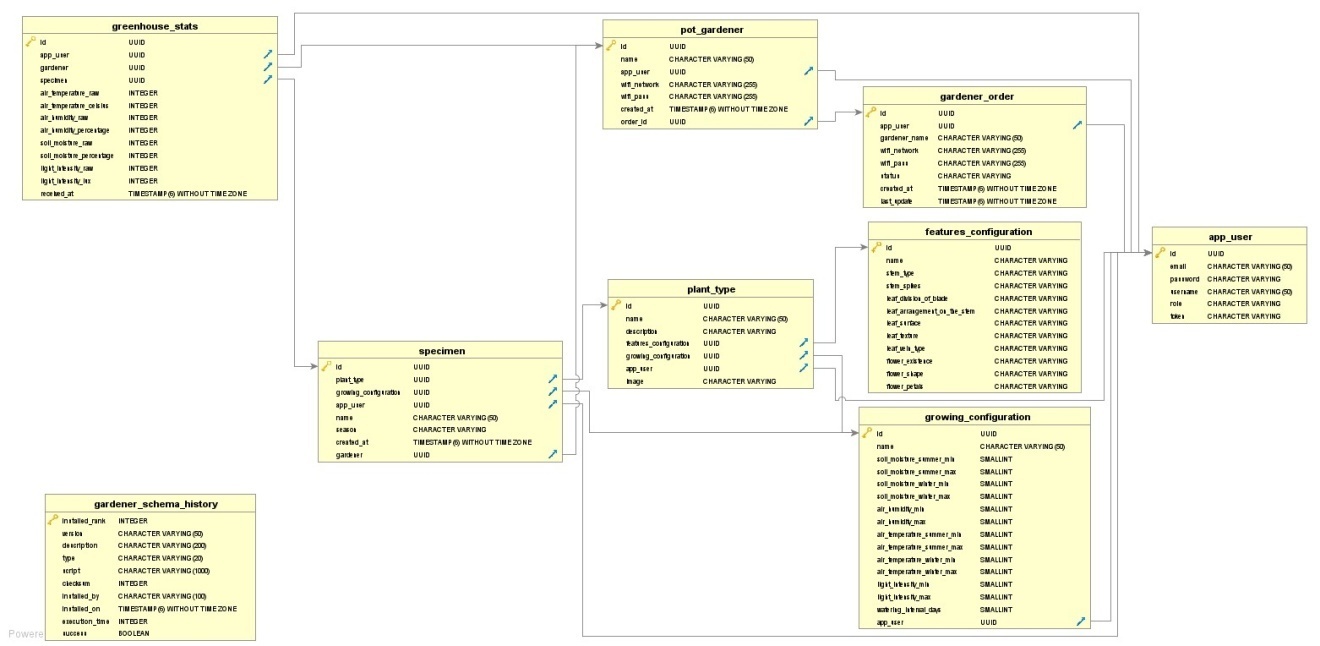


Figura . Diagrama relațională a entităților din baza de date

Se observă un tabel „gardener\_schema\_history”. Acesta nu conține date despre aplicația în sine ci despre migrările care au adus baza de date în stadiul actual. Aceste migrări sunt practic pașii parcurși pentru ca baza de date să ajungă în această stare.

### Flyway

Pentru construirea bazei de date am folosit sistemul de versionare a bazelor de date Flyway[[39]](#footnote-40). Scopul utilizării unui sistem de versionare a bazelor de date este să asigure consistență bazei de date în orice mediu de dezvoltare sau execuție. El este compatibil cu Spring și ușor de configurat în fișierul application.properties. În figura 5.x sunt vizibile configurările pe care le-am făcut pentru baza noastră de date.

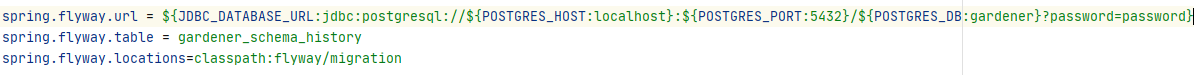


Figura . Configurarea Flyway

Flyway ne permite să scriem în fișiere separate cod sql care va fi rulat pe baza de date. O astfel de acțiune se numește rularea unei migrări. Flyway păstrează un tabel cu o istorie a tuturor migrărilor executate pe baza de date. Numele standard al acestui tabel e „flyway\_schema\_history”, în cazul nostru l-am numit „gardener\_schema\_history” (vezi figura 5.x). Cu ajutorul acestui tabel, Flyway știe când o migrare nouă a fost adăugată și nu a fost executată și o să încerce să o execute. Când o migrare este executată cu succes, această informație e salvată în „schema\_history”. În figura 5.x se pot observa primele migrări pe care le-am executat în sistemul nostru.

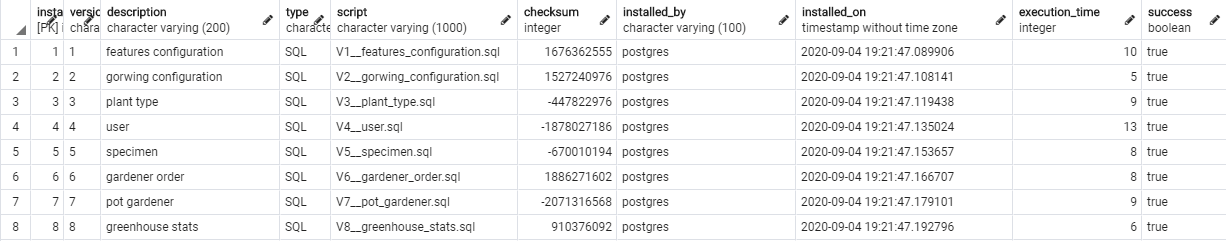


Figura . Migrări executate pe baza de date a sistemului

De asemenea, Flyway execută și o verificare a integrității migrărilor deja rulate. El verifică checksum-ul fișierelor actuale cu cel salvat când migrarea a fost executată cu succes. Dacă ele diferă, Flyway aruncă erori de consistență. Astfel, pentru fiecare modificare a bazei de date vom adăuga un nou fișier cu cod SQL ce va fi migrat. În funcție de numele fișierului Flyway știe în ce ordine trebuie executate migrările. Standardul e prefixul „V” urmat de numărul migrării și „\_\_” (de două ori caracterul underscore) ex: V4\_\_user.sql.

Astfel, utilizând Flyway suntem asigurați de consistența bazei de date și în același timp constrânși să o respectăm în continuare.

## Aplicația client

Aplicația client este responsabilă să ofere o interfață grafică utilizatorului și să comunice cu aplicație server. Ea este implementată ca o aplicație web și este dezvoltată în framework-ul Angular 10.

O aplicație web este o aplicație software care rulează pe un server web. Am ales această opțiune deoarece, spre deosebire de aplicațiile desktop/mobile, care sunt stocate pe un dispozitiv, aceasta este accesibilă de pe orice dispozitiv (desktop sau mobile) cu o conexiune la internet.

Angular este un farmework de JavaScript open-source pentru aplicatiile web prezentate ca o singura pagina dinamica (single page applications)[[40]](#footnote-41). Angular este bazat pe TypeScript. În plus față de JavaScript, TypeScript, după cum îi spune numele asigură consistența tipului datelor, astfel oferind un cadru mai ușor de dezvoltare.

Arhitectura pe care este bazat framework-ul Angular este MVC (Model View Controller). După cum se observă și în figura 5.x, aceasta înseamnă că modelul este modificat de către controller în funcție de modificările utilizatorului din nivelul view. De asemenea, view-ul este modificat de căter controller în funcție de starea modelului.

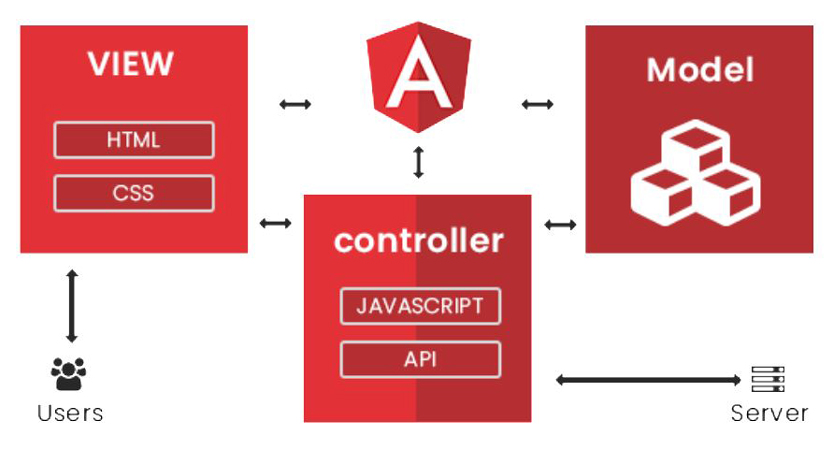


Figura . Arhitectura MVC în Angular [[41]](#footnote-42)

O trăsătură cheie a Angular sunt componentele. Un component este unitatea de bază pe care este bazată dezvoltarea unei aplicații Angular. Un component poate sa agregheze mai multe alte componente. Practic, o aplicație Angular este un arbore de componente.

### Componentele aplicației

În figura 5.x putem observa structura componentelor din aplicația noastră de grădinărit. Componentele cu violet fundal sunt vizibile oricărui utilizator care accesează aplicația web. Componentele cu fundal verde sunt disponibile doar dacă utilizatorul este autentificat în aplicație. Componetele cu fundal roșu sunt disponinibile doar utilizatorului „Administrator” când acesta este autentificat în aplicație. Observăm o există un componentă rădăcină <gar-root-app>. Practic conținutul acesteia este randat în fișierul index.html. Ea întotdeauna va afișa componentele: <gar-header> și <gar-footer> împreună cu conținutul componentei <router-outlet>. Componenta <router-outlet> va afișa conținutul decis de modulul de routing al Angular. Deciziile acestea se fac în funcție de specificațiile de tip cheie – valoare din componenta de routing.

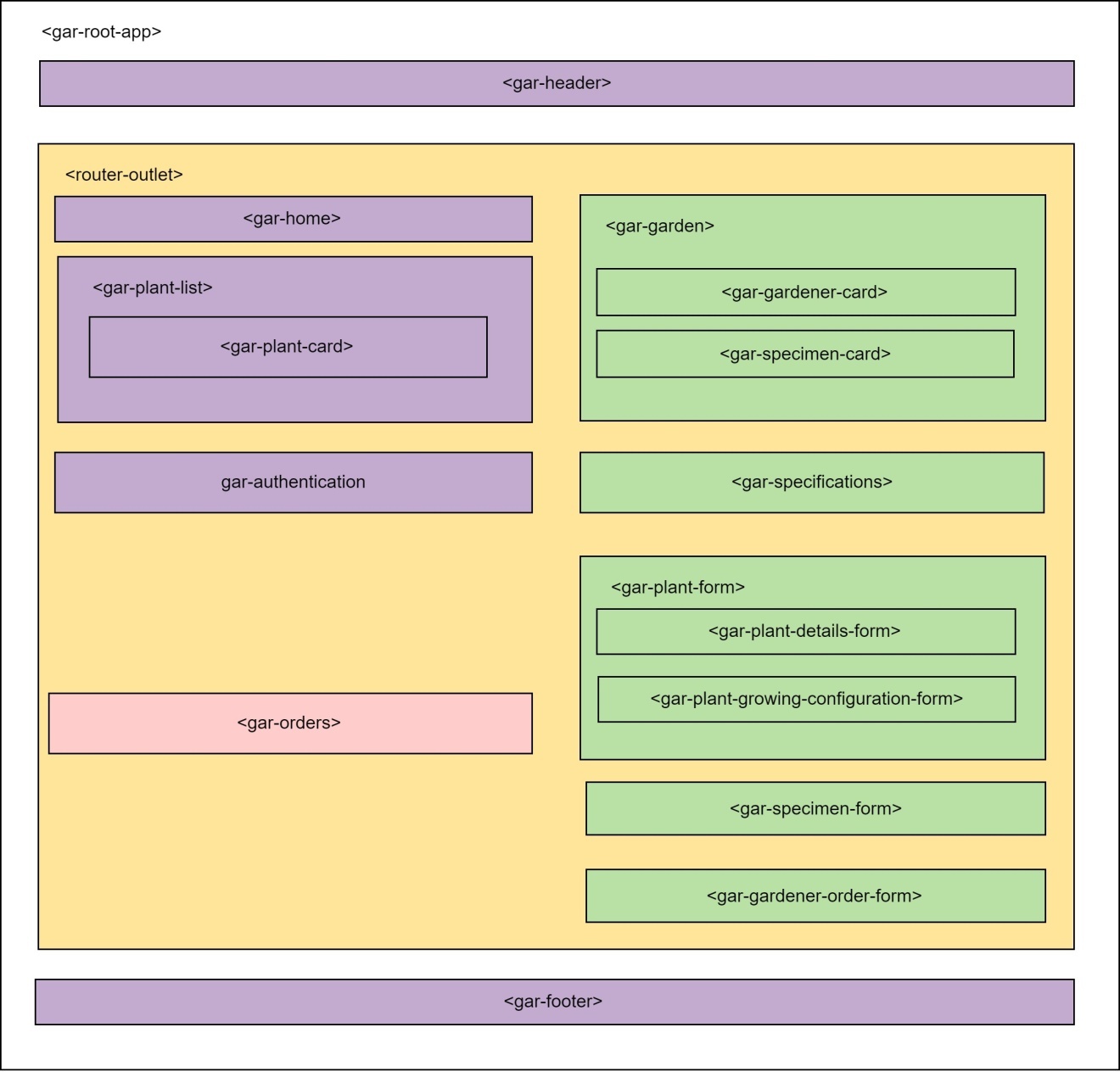


Figura . Componentele aplicației client

### OpenAPI, npm și Verdaccio

Pentru generarea API-ului pentru modulul client ne-am folosit de asemenea de fișierul YAML ce conține specificațiile API-ului nostru în formatul OpenAPI Specification și de OpenAPI Generator. De această dată am folosit generatorul typescript-angular[[42]](#footnote-43) pentru a genera un client API. Folosind managerul de pachete npm[[43]](#footnote-44) am instalat dependențele acestui client și am creat un pachet de npm au API-ul nostru.

În figura 5.x se poate observa în stânga aceeași specificație OpenAPI pe care am folosit-o în aplicația server iar în dreapta, de această dată este codul generat pentru aplicația client.



Figura . Specificație OpenAPI (stânga) și codul generat de generatorul typescript-angular (dreapta)

Pentru utilizarea locală a acestui pachet am ales să îl publicăm într-un registry local utilizând aplicația Verdaccio[[44]](#footnote-45). Verdaccio oferă posibilitatea publicării pachetelor npm într-un registry privat, local.

Pentru instalarea acestui pachet am modificat fișierul .nprc al aplicație Angular astfel încât atunci când npm instalează dependențele aplicației, pentru pachetele din modulul nostru să acceseze registry-ul local, nu cel oficial de la npm.



Figura . Modificarea fișierului .npmrc pentru utilizarea registry-ului local oferit de Verdaccio

### Notifications API

<<Poate\_Urmează\_până\_luni\_dimineață >>

## Modulul Hardware

Modulul hardware reprezintă partea din sistemul de grădinărit care se ocupă în principal cu monitoarizarea plantei și controlarea mediului acesteia pentru o dezvoltare propice dar și cu informarea utilizatorului cu privire la starea mediului. Pentru ca acest modul să își atingă scopul de a furniza date utilizatorului (dar și de a menține umiditatea solului în parametrii specificați de utilizator) el trebuie să comunice cu aplicația server. În figura 5.x observăm comunicarea bidirecțională dintre aplicația server și modulul hardware. Modulul hardware furnizează aplicației server valorile citite de la senzori. Aplicația server transmite valoarea minimă pentru umiditatea solului.

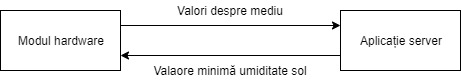


Figura . Comunicarea între modulul hardware și aplicația sever

Conform concluziei capitolului 3, pentru sistemul de control din modulul hardware utilizăm o plăcuța R3 Uno +WiFi. Pentru dezvoltarea codului necesar acestui modul am folosit limbajul de programarea Arduino și mediu de dezvoltare Arduino Software[[45]](#footnote-46). Ținem să amintim arhitectura plăcuței R3 Uno WiFi este compusă din modulul ATmega382P și modulul ESP8266. Aceste module sunt programabile independent dar pot comunica între ele. Plăcuța R3 UNO + WiFi poate funcționa 5 moduri distincte. Ele sunt evidențiate în figura 5.x. Pentru a selecta unul dintre acele moduri, trebuie introdusă configurația respectivă pe cele 8 switch-uri ale plăcuței. Modul de programare ATmega382P, de exemplu este selectat prin setarea switch-urilor în pozițiile (OFF, OFF, ON, ON, OFF, OFF, OFF, OFF).

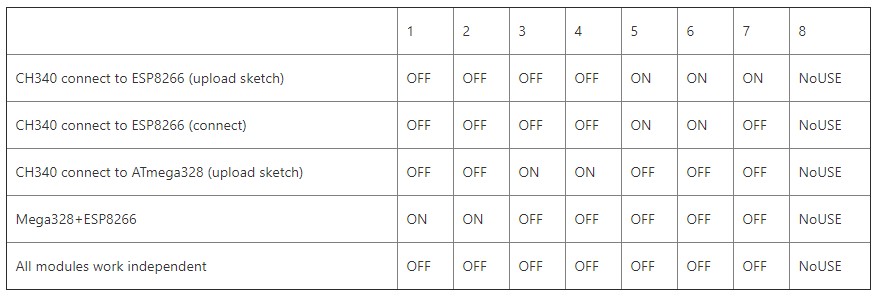


Figura .4 Moduri funcționare plăcuță R3 Uno + WiFi

Vom împărți modulul nostru hardware în două submodule mai mici în funcție de responsabilitățile acestora. Primul este „Modulul de grădinărit”. Acesta este responsabil pentru citirea datelor de la senzori și acționarea pompei. Al doilea este „Modulul de comunicare”. El este format din modulul ESP8266 și este responsabil cu comunicarea cu Modulul de grădinărit și cu aplicația server. În figura 5.x se observă cum am împărțit aceste responsabilități și cum comunică modulele între ele.

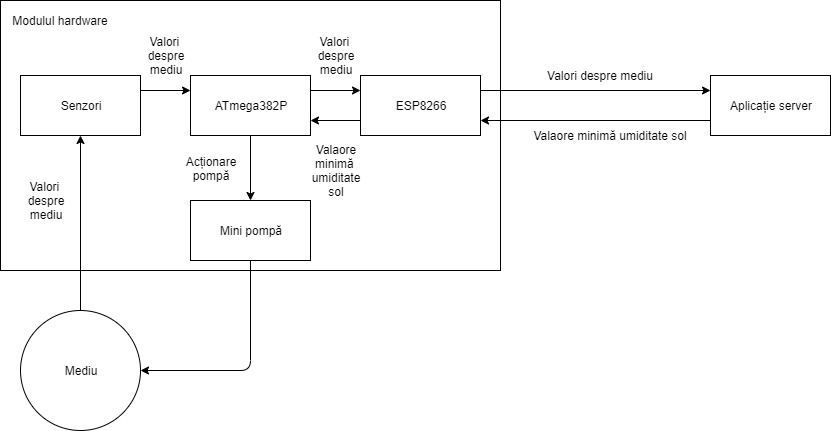


Figura .5 Modul hardware – Responsabilități

În continuare, vom prezenta în două subcapitole fiecare dintre responsabilitățile și implementarea celor două submodule. Un al trei-lea subcapitol va prezenta modul acestor de comunicare între ele.

### Submodulul de grădinărit – ATmega382P

După cum am prezentat în subcapitolul 2.1.8, datele de care suntem interesați să le cunoaștem despre mediu sunt: temperatura aerului, umiditatea aerului, temperatura solului și intensitatea luminoasă a mediului. Pentru a putea măsura aceste valori am întrebuințat senzori compatibili cu R3 Uno + WiFi.

Pentru a măsura temperatura și umiditatea aerului am ales să folosim senzorul DHT11. Acesta este un senzor cu funcție dublă, care poate măsura atât temperatura aerului, cât și umiditatea din el. Am ales acest senzor și din considerente financiare. În figura 5.x se poate observa senzorul DHT11. Acest senzor poate măsura temperaturi între 0 și 50 de grade Celsius. Umiditatea aerului este măsurată de acest senzor în valori de la 0 la 100, unde 100 înseamnă ploaie[[46]](#footnote-47).

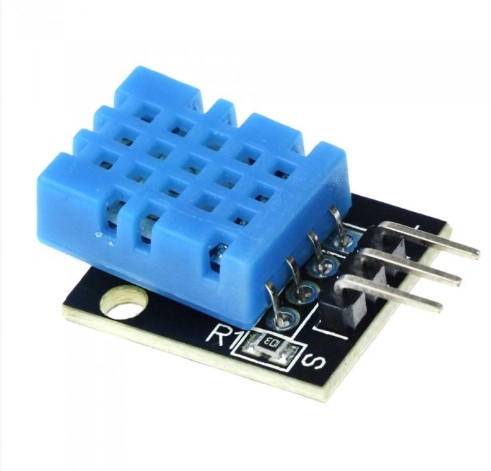


Figura . Senzor DHT11[[47]](#footnote-48)

Pentru a măsura umiditatea solului am ales un modul compus din senzorul FC28 și cipul LM393. Acestea pot fi observate în figura 5.x. Cele două plăcuțe expuse ale senzorului analogic funcționează ca sonde, acționând ca un rezistor variabil. Cu cât este mai multă apă în sol cu atât conductivitatea dintre plăcuțe va fi mai mare și rezistența mai mică. Senzorul înregistrează valori între 0 și 1023[[48]](#footnote-49).

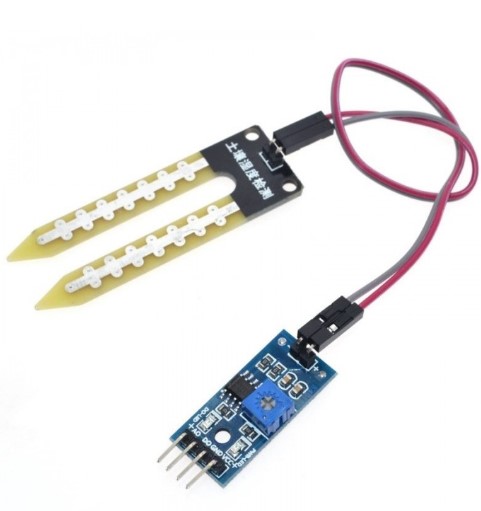


Figura . Modul cu senzor de umiditate sol FC28 și cip LM393[[49]](#footnote-50)

Măsurarea intensității luminoase a mediului o vom face folosind un fotorezistor și o rezistență fixă. Un fotorezistor (LDR - Light Decreasing Resistance) este un element fotoconductor. Aceasta înseamnă că această componentă electronică își crește conductivitatea (pierde din rezistență) cu cât este expus la o lumină mai puternică. În figura 5.x observăm fotorezistorul folosit. Pentru a putea măsura cu cât fotorezistorul devine mai mult sau mai puțin conductiv, folosim o rezistenă fixă pentru a crea un divizor rezistiv.[[50]](#footnote-51)



Figura . Fotorezistor[[51]](#footnote-52)

Ulterior citrii datelor din mediu, vom modifica mediul în funcție de configurația plantei. După cum am prezentat în capitoulul 4, singurul element pe care îl vom modifica este umiditatea solului. Pentru aceasta am ales să utilizăm o mini pompă submersibilă. Deoarece am dorit alimentarea mini-pompei de la o altă sursă de tensiune, și nu din plăcuță, am utilizat și un releu în combinație cu aceasta. În figura 5.x observăm în stânga mini pompa submersibilă iar în dreapta un modul de releu de 5V.

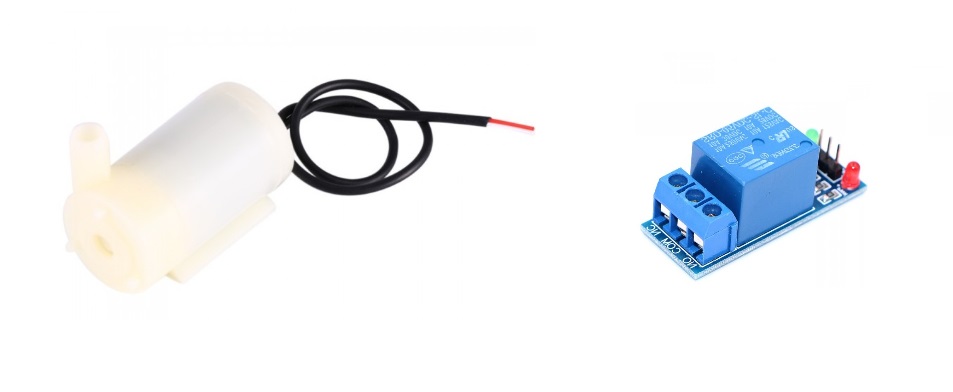


Figura . Mini pompa submersibilă[[52]](#footnote-53) și Modul releu de 5V[[53]](#footnote-54)

După cum se observă și în figura 5.x pentru a citi informațiile furnizate de acești senzori și pentru controla pompa de apă avem nevoie de modul „CH340 connect to ATmega328” al plăcuței. Acesta este modul în care plăcuța se comportă ca un Arduino Uno.

Pentru a obține date de la senzori, îi conectăm la pinii analogi ai cipului ATmega382P, setăm acești pini ca pini de intrare și citim valorile prin funcția specifică Arduino „analogRead”.

Pentru a controla mini pompa de apă, conectăm intrarea acesteia la un pin digital al ATmega382P și scriem pe acesta valori „HIGH” sau „LOW”, cu funcția specifică Arduino „digitalWrite”.

Din cauza sensibilității senzorului de temperatură și umiditate DHT11 avem nevoie să folosim și timpi morți în cod pentru a nu citi valori eronate.

### Submodulul de comunicare – ESP8266

Funcția de comunicare, atât cu aplicația server cât și cu modulul de grădinărit îi revine modului de comunicare. Pentru implementarea acestui modul vom scrie cod, de asemenea, în Arduino Software dar de data aceasta vom încărca codul pe modulul ESP8266. Pentru a aduce plăcuța în starea în care codul poate fi încărcat pe ESP8266, aceasta trebuie să fie în starea „CH340 connect to ESP8266 (upload sketch)”. Pentru aceasta switch-urile plăcuței trebuie să fie în starea (OFF, OFF, OFF, OFF, ON, ON, ON, OFF) conform figurii 5.x.

Modulul de comunicare execută următoare secvență de pași

* Se conectează la rețeaua de WiFi. Credențialele pentru aceasta trebuie specificate în codul sursă. Credențialele se referă la numele rețelei de WiFi și parola acesteia.
* Se conectează la aplicația server. Adresa aplicație server trebuie specificată în codul sursă.
* Așteaptă valorile transmise de modulul de grădinărit
* Citește valorile transmise de modulul de grădinărit
* Face un apel HTTP de tipul POST cu valorile citite către aplicația server
* Așteaptă maximum 5 secunde răspunsul de la aplicația server. În caz că aplicația server nu răspunde în 5 secunde, se reîncearcă conectarea la server.
* Primește răspunsul de la aplicația server. Răspunsul constă în valorile de configurare pentru modulul de grădinărit
* Procesează răspunsul de la aplicația server. De exemplu elimină spațiile din răspuns și caracterul de rând nou
* Trimite răspunsul procesat către modulul de grădinărit
* Începe de la primul pas.

Pentru a putea încărca acest cod pe plăcuță a trebuit sa importăm un manager de plăcuță specific ESP8266[[54]](#footnote-55) și să selectăm plăcuța „Generic ESP8266 Module” cu viteză de încărcare de 115200, flash size de 512K și flash mode „DIO”.

### Comunicare dintre submodulele modulului hardware

Pentru ca modulul ATmega382P și modulul ESP8266 să poată comunica între ele, plăcuța trebuie să fie în starea „Mega328+ESP8266”, ceea ce înseamnă conform figurii 5.x că switch-urile acesteia trebuie să fie în starea (ON, ON, OFF, OFF, OFF, OFF, OFF, OFF).

Modulele comunică între ele prin circuite fizice. Fiecare modul are o componentă UART (Asynchronous Receiver/Transmitter) care este capabilă să transmită și să primească date. Componentele UART au două canale de comunicare. Unul pentru transmiterea datelor: Tx, și unul pentru recepționarea datelor: Rx[[55]](#footnote-56). În figura 5.x se observă cum sunt două componente UART conectate între ele pentru a putea comunica. Componentele UART transmit datele în mod asincron

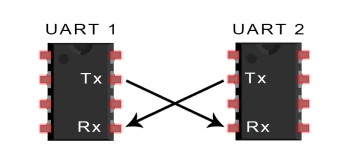


Figura . Comunicare între două componente UART

Comunicarea între cele două module se face prin funcția Arduino „Serial” care trebuie inițializată la aceeași viteză în amble module.

Metoda pe care am ales-o pentru a implementa comunicarea dintre submodulul de grădinărit și submodulul de comunicare urmează pașii în buclă infinită:

* Submodulul de comunicare așteaptă un mesaj de la submodulul de grădinărit prin UART
* Submodulul de grădinărit trimite datele citite de la senzori sub formă de JSON prin UART
* Submodulul de grădinărit așteaptă 10 secunde
* Submodulul de comunicare trimite un request de tipul POST către aplicația server
* Submodulul de comunicare primește răspunsul de la aplicația server
* Submodulul de comunicare transmite răspunsul către submodulul de grădinărit prin UART
* Submodulul de grădinărit citește mesajul prin UART

## Alte instrumente

### Git

De menționat că în dezvoltarea sistemului de grădinărit am folosit aplicația de versionare git[[56]](#footnote-57).

### IntelliJ IDEA

### Arduino Software (IDE)

# Testare şi Validare

<< intro tipuri de testare and shit>>

Aproximativ 5% din total.

## Testarea componentelor

Testarea componentelor are ca scop confirmarea faptului că piesele folosite pentru dezvoltarea unui sistem funcționează în parametrii specificați. Pentru partea harware a sistemului de grădinărit am testat toate piesele componente în mod individual înainte de a începe asamblarea modului hardware.

### Testarea plăcuței R3 UNO + WiFi

Pentru testarea plăcuței R3 UNO + WiFi am validat patru cazuri de test separate:

* Testarea conexiunii plăcuței cu laptopul
* Testarea modulului ATmega328P
* Testarea modulului ESP8266
* Testarea comunicării dintre modulul ATmega328P și ESP8266

Pentru testarea conexiunii plăcuței cu laptopul scenariul pe care l-am urmat a fost ca după conectarea plăcuței la laptop prin cablul micro USB, am căutat că aceasta întradevăr e recunoscută și apare în dispozitivele periferice vizibile de către Managerul de Dispozitive ale sistemului de operare. În cazul în care aceasta nu apare ca dispozitiv periferic, poate fi o problemă cu plăcuța în sine sau cu cablul micro USB folosit. Există cabluri micro USB doar pentru încărcare, care nu folosesc pinii de transmitere a datelor sau există plăcuțe care sunt defecte din fabricație.

Pentru testarea modulului ATmega328P am avut nevoie să verificăm dacă acesta funcționează independent. Pentru aceasta a fost nevoie să setăm plăcuța în modul de operare „CH340 connect to ATmega328 (upload sketch)” și să selectăm tipul de plăcuță „Arduino Uno” în Arduino Software, după cum se observă și în figura 6.x.

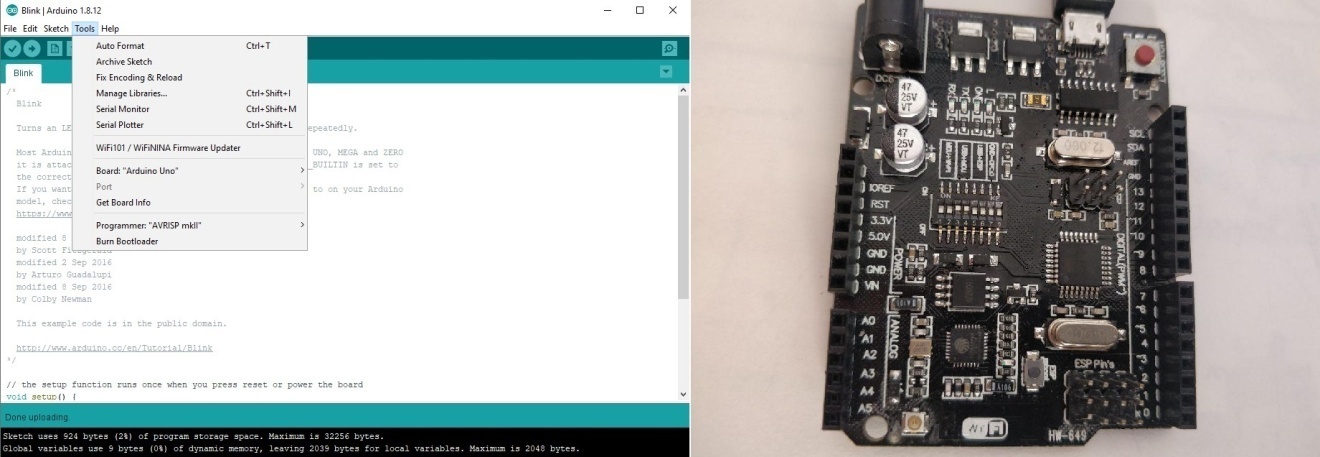


Figura . Setările pentru utilizare ATmega328P

Apoi, am ales un exemplu de cod sursă din biblioteca Arduino. Exemplul ales a fot „Blink”[[57]](#footnote-58). Acesta făcea ledul încorporat al plăcuței să se stingă și să se aprindă la un interval regulat. Observăm în imaginea 6.x plăcuță în cu ledul în stare stinsă în stânga și cu ledul în stare aprinsă în dreapta.

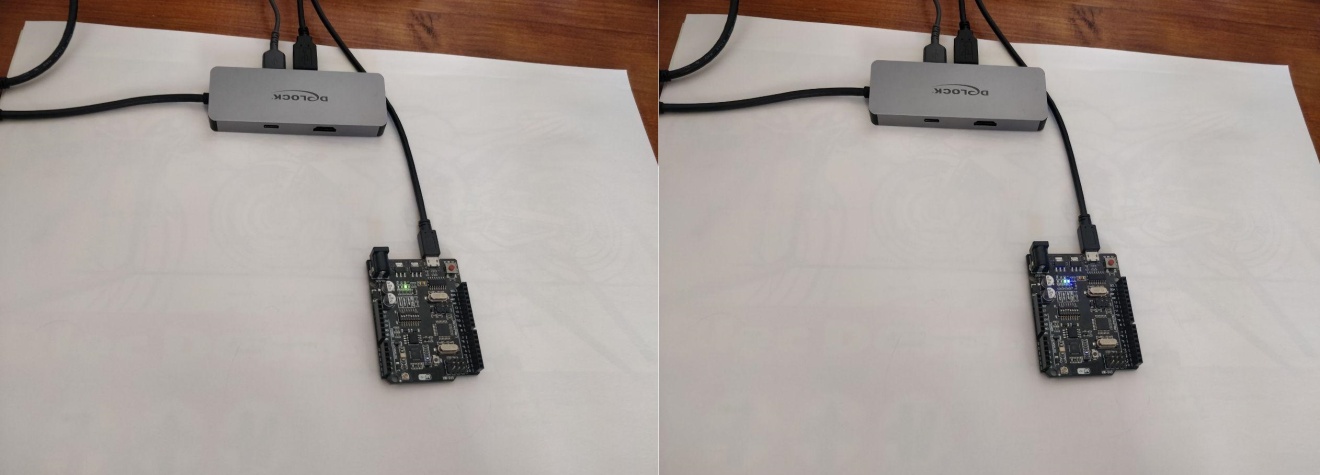


Figura .2 Testare ATmega328P

Al treilea scenariu de test făcut în testarea plăcuței a fost testarea modulului ESP8266. Pentru aceasta a fost nevoie să setăm plăcuța în modul de operare „CH340 connect to ESP8266 (upload sketch)” și să selectăm tipul de plăcuță „Generic ESP8266 Module” în Arduino Software, după cum se observă și în figura 6.x. Pentru a avea acces la acest tip de plăcuță a trebuit în prealabil să importăm un manager de plăcuță specific ESP8266[[58]](#footnote-59).

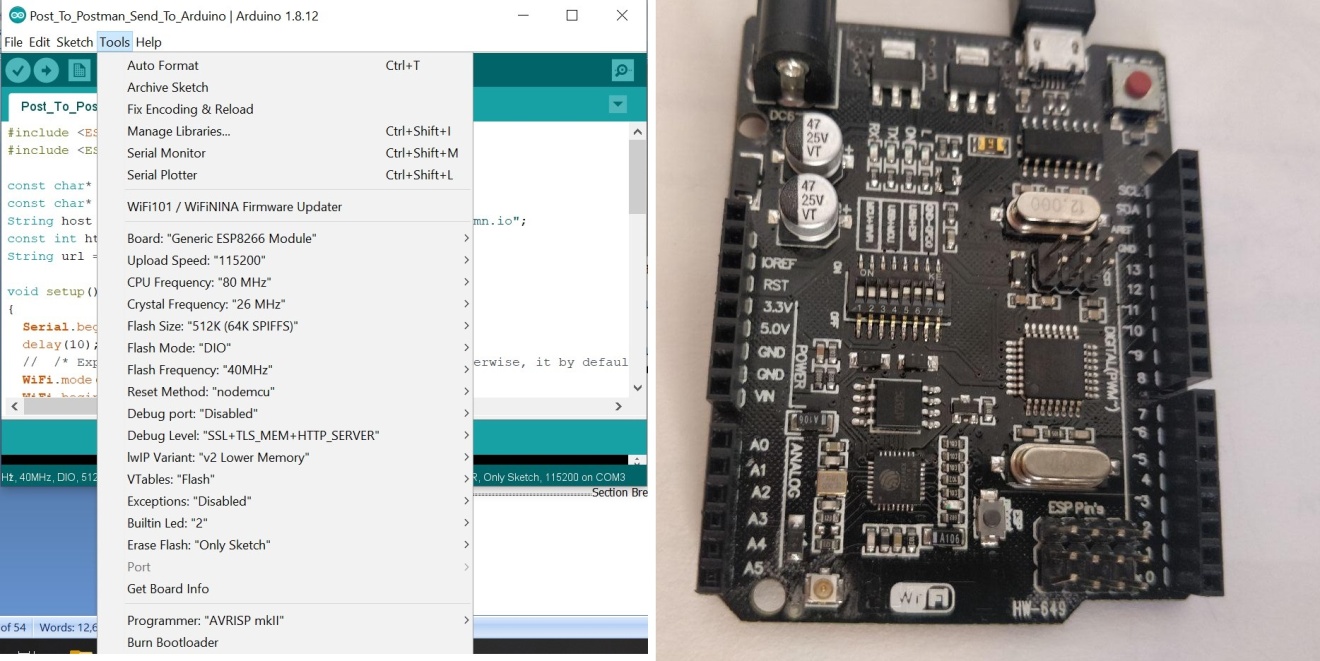


Figura . Setările pentru utilizare ESP2866

Pentru testarea modulului am folosit același cod din schița anterioară, „Blink” doar că l-am adaptat pentru un led legat la pinul 2 al modulului ESP8266. În figura 6.x se observă atât starea în care ledul este oprit în stânga, cât si starea în care ledul este aprins, conform codului.

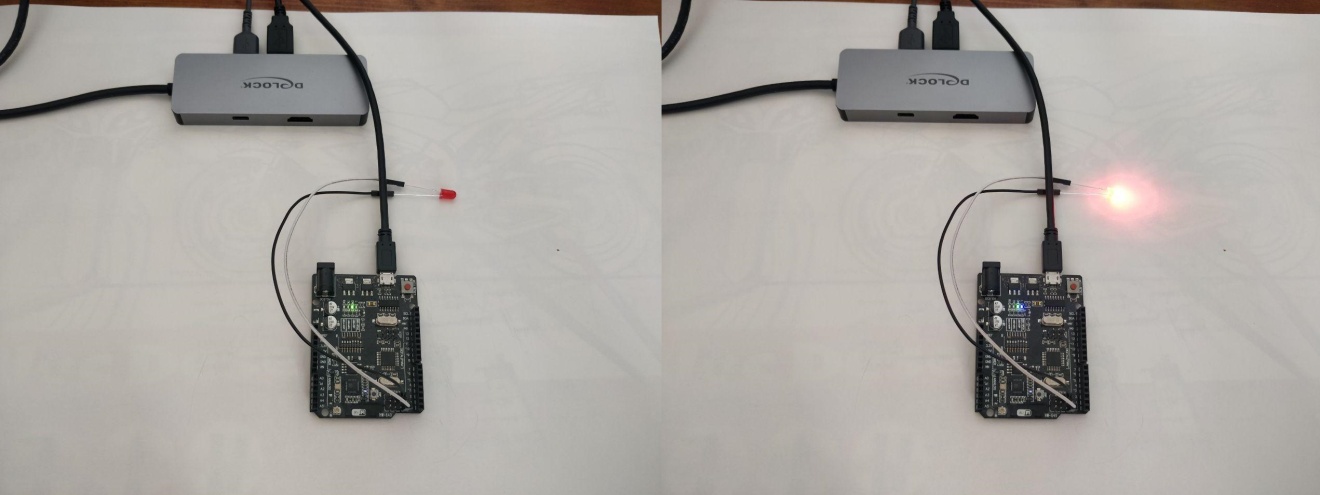


Figura . Testare ESP8266

Pentru acest modul am testat și faptul că se poate conecta la internet. Pentru aceasta am folosit aplicația Postman unde am creat un „mock server” (un serviciu oferit de Postman prin care putem crea un server pe care mai apoi îl configurăm să răspundă în funcție de anumite exemple pe care noi le creăm[[59]](#footnote-60)).

Scenariul a fost să trimitem un apel http de tipul POST către serverul creat și să afișăm răspunsul de la acesta. În figura 6.4 se poate observa atât serverul creat în Postman cât și ce afișează modulul ESP8266 prin comunicarea serială.

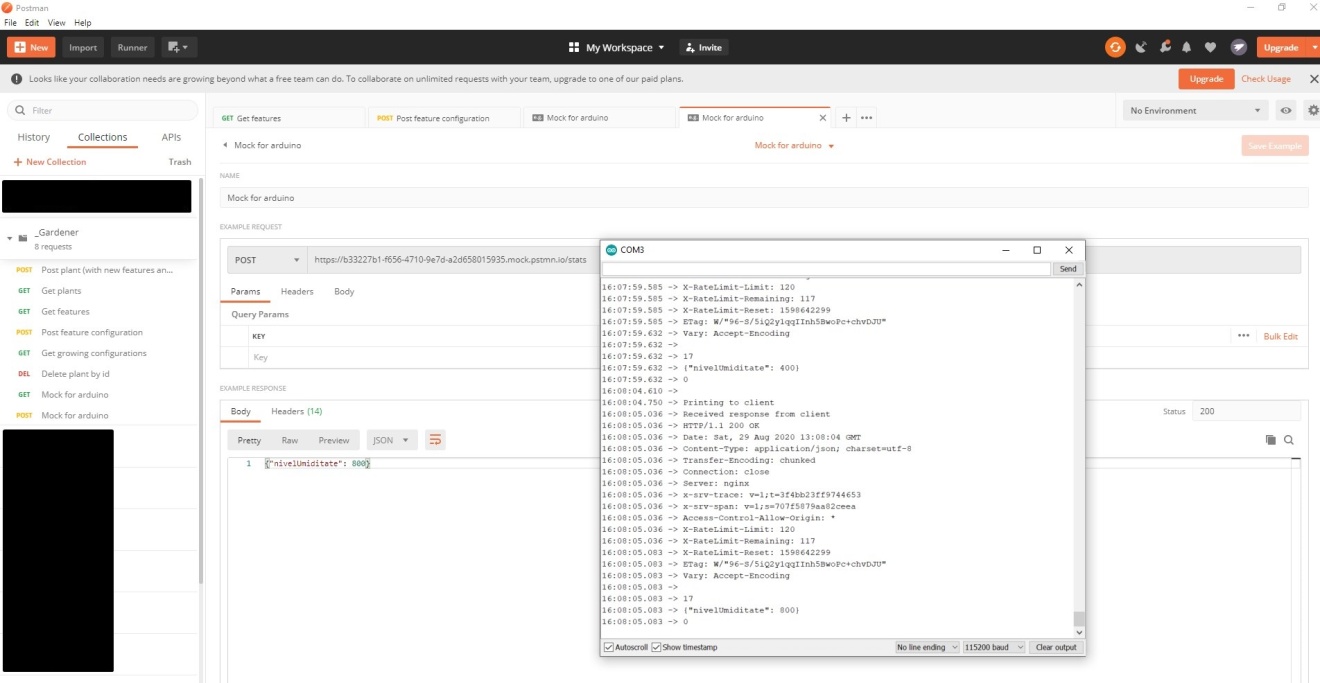


Figura . Serve mock Postman și Monitorul serial Arduino

Pentru testarea comunicării dintre modulele ATmega328P și ESP8266 am folosit și un senzor de umiditate și temperatură, DHT11. Senzorul a fost testat individual în prealabil. Procesul și rezultatele scenriului de test pentru senzor sunt prezentate în subcapitolul 6.1.2. Testarea senzoruluiDHT11. Recomandăm parcurgerea acelui capitol și apoi revenirea la acesta.

Pentru testarea comunicării dintre cele două module am ales următorul scenariu: am conectat senzorul de umiditate și temperatură la un pin analog al ATmega328P și am transmis prin modul de comunicare serial valorile de temperatură citite de senzor, iar în modulul de ESP8266 am citit datele primite prin comunicarea seială, urmând ca atunci când aceste valori depășeau pragul de 30 de grade, să aprindem un led conectat la un pin digital al modului ESP8266.

Pașii urmați pentru implementarea acestui scenariu au fost:

* Setarea plăcuței în modul „CH340 connect to ATmega328 (upload sketch)”
* Încărcarea codului care citește temperatura de la senzor și o transmite serial
* Setarea plăcuței în modul „CH340 connect to ESP8266 (upload sketch)” și resetarea acesteia (apăsarea butonului de reset sau deconectarea urmată de reconectarea plăcuței la laptop)
* Încărcarea codului care citește valorile transmise prin comunicarea serială și aprinde ledul dacă valoarea primită e mai mare de 30
* Setarea plăcuței în modul „Mega328+ESP8266”

Ultimul pas este vizibil în figura 6.4. Când plăcuța este în acest mod de operare ea . rulează simultan codul de pe cele două module și conectează componentele UART a acestor între ele ca acestea să poată comunica serial.

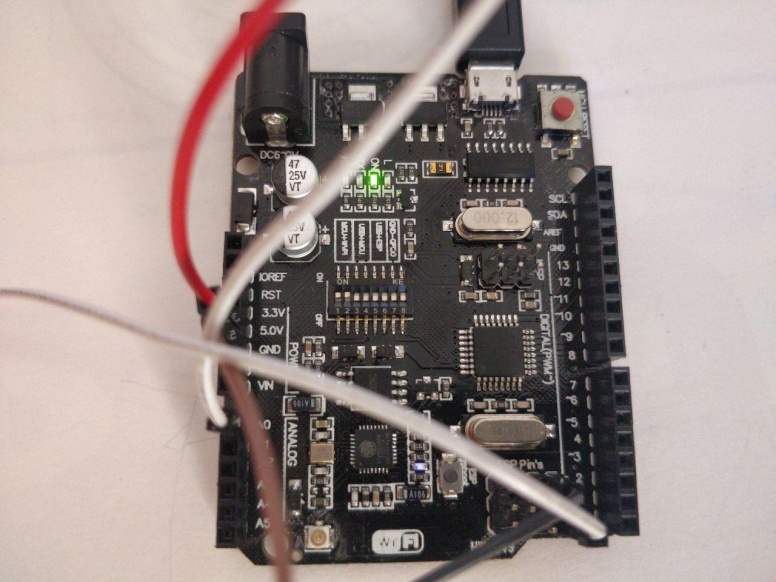


Figura . Plăcuța setată în modul de operare „Mega328+ESP8266”

Pentru validarea scenariului am crescut temperatura înregistrată de senzorul conectat la ATmega328P cu ajutorul unui feon. După câteva secunde, led conecta la modulul ESP8266 s-a aprins (figura 6.5 stânga). Pentru a scădea temperatura înregistrată de senzor am pulverizat apă rece pe acesta. După câteva secunde ledul a început să pâlpâie și apoi s-a stins (figura 6.5 dreapta). Pâlpâirea ledului a însemnat că acesta a înregistrat valori alternativ mai mici și mai mari de 30 de grade pentru câteva secunde, iar stingerea sa că a înregistrat doar valori mai mici de 30 de grade. Fluctuații mici de temperatură sunt de aștepta, astfel că am considerat testul executat cu succes.

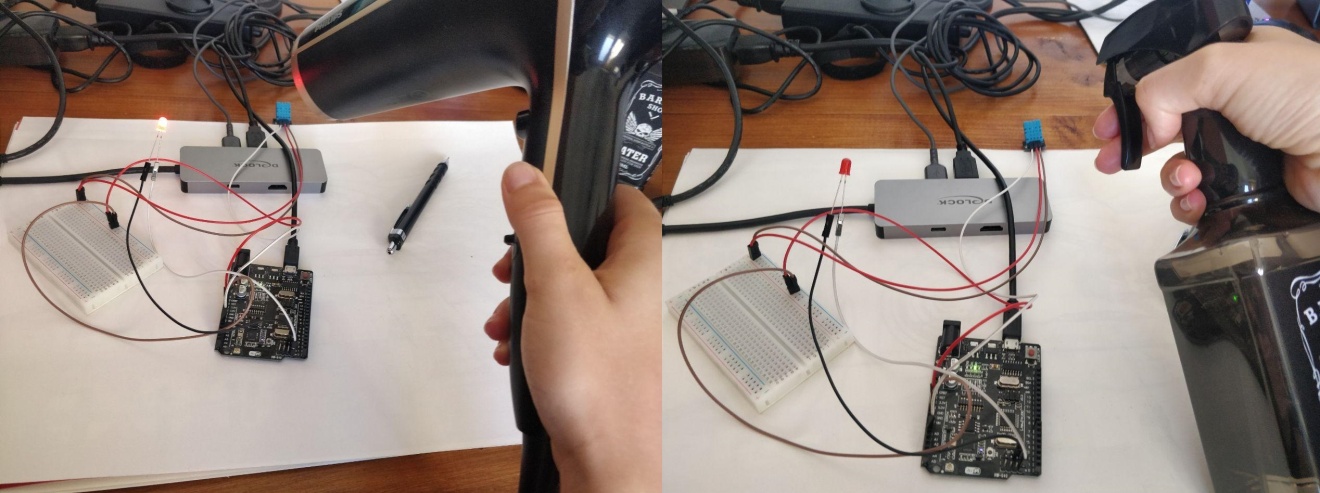


Figura . Rezultatele testării comunicării dintre modulul Atmeg328P și ESP8266

Conform rezultatelor obținute în urma acestor scenarii de testarea am concluzionat că plăcuță R3 UNO +WiFi funcționează corect și este pregătită să susțină implementarea proiectului.

### Testarea senzorului DHT11

Senzorul DHT11 (figura 5.x) este un senzor care măsoară temperatura și umiditatea din aer. Am conectat acest senzor la plăcuța R3 Uno + WiFi conform unui tutorial prezentat de[[60]](#footnote-61) în care este specificat că avem nevoie și de o bibliotecă adițională[[61]](#footnote-62) pentru a putea citi datele de la DHT11.

Scenariul de test pe care l-am folosit a fost să printăm valorile înregistrare de senzor pe canalul serial, să modificăm starea mediului din exterior și să observăm în fereastra „Serial Monitor” a mediului de dezvoltare ce valori sunt înregistrare. De menționat că rata de transfer setată în codul sursă trebuie să fie acceași cu rata de transfer a ferestrei de comunicare.

Observăm în figura 6.6 că în momentul în care pulverizăm apă pe senzor, acesta înregistrează o termperatură din ce în ce mai scăzută și o umiditate a aerului din ce în ce mai ridicată.



Figura . Măsurători înregistrare de senzorul DHT11 sub acțiunea umezirii

Pentru a testa și că senzorul își revine din această stare am accelerat procesul prin utilizarea unui feon. În figura 6.7 se observă cum senzorul înregistrează temperaturi din ce în ce mai ridicate și o umiditate a aerului tot mai scăzută.

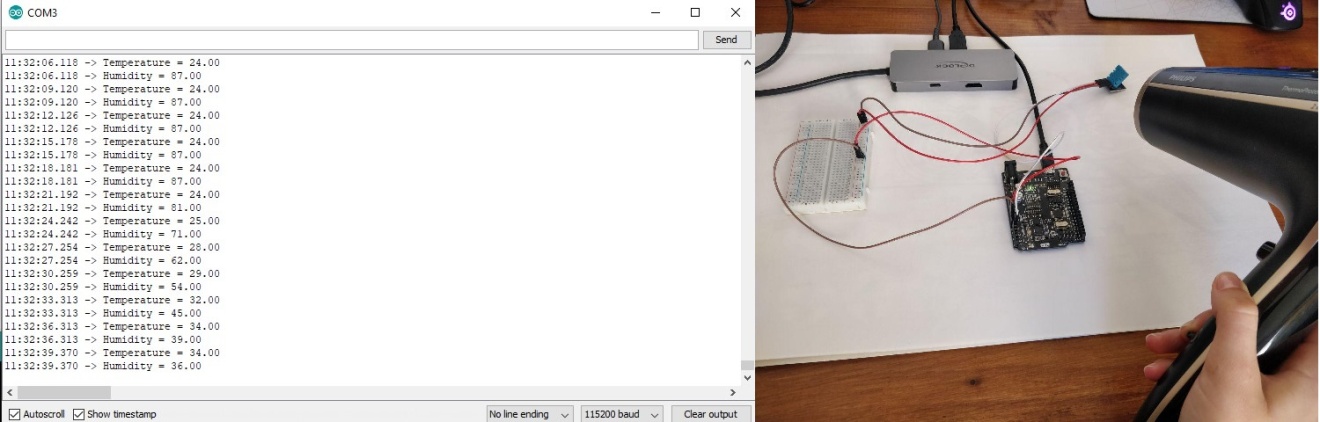


Figura . Măsurători înregistrarea de senzorul DHT11 sub acțiunea încălzirii

Conform rezultatelor obținute în urma acestui scenariu de testarea am concluzionat că senzorul DHT11 pentru măsurarea temperaturii și a umiditătii aerului funcționează în mod adecvat.

### Testarea modulului de măsurarea a umidității solului

Pentru testarea modulului de măsurare a umidității solului am abordat aceeași metodă ca în subcapitolul 6.1.2. și anume înregistrarea valorilor citite de senzor și afișarea acestora în pe monitorul serial.

În figura 6.8 dreapta observăm senzorul de umiditate a solului introdus într-un recipient cu un pământ extrem de uscat în care nu ar putea nicio plantă să se dezvolte. Valorile citite de sensor în acest mediu sunt expuse în figura 6.8 stânga. Observăm că doarece senzorul este într-un mediu cu o conductivitate extrem de scăzută, înregistrăm valori aproape maxime ale rezistenței.

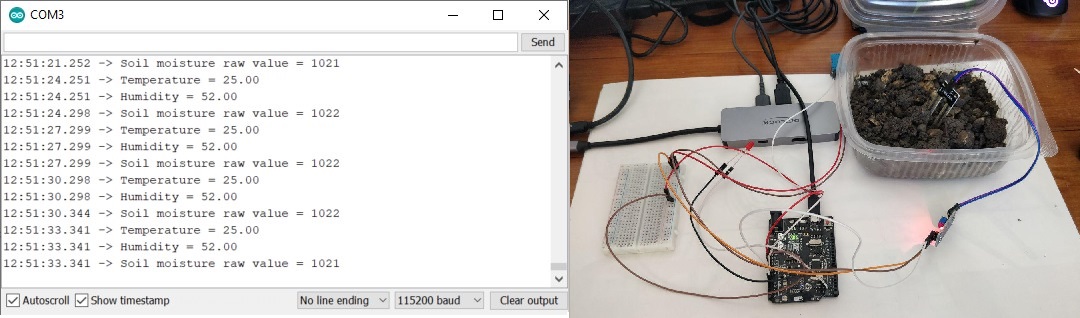


Figura . Măsurători înregistrate de modulul de umiditate a solului într-un mediu uscat

În figura 6.9 am introdus senzorul de umiditate în recipientul în care creștem un avocado din sâmbure. Observăm în partea dreaptă a figurii că senzorul este introdus aproape total în apă. În partea stângă a figurii 6.9 sunt prezentate valorile înregistrate de senzor în acest mediu. Fiind un mediu cu o conductivitate mare, modulul înregistrează valori ale rezistenței sub jumătate ca în cazul anterior.

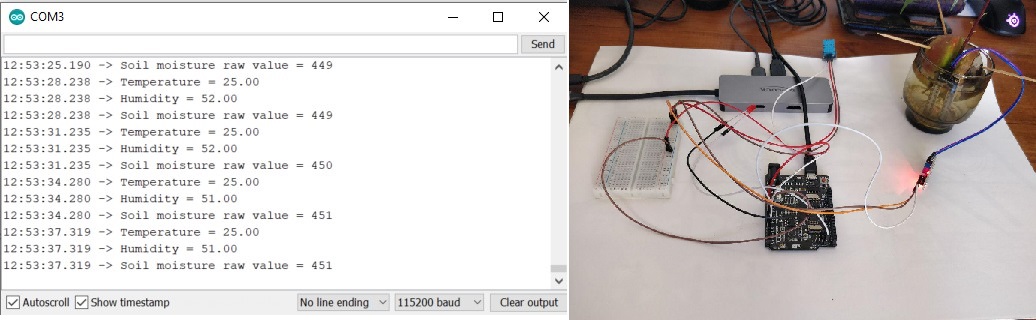


Figura . Măsurători înregistrare de modulul de umiditate a solului într-un mediu umed

Conform rezultatelor obținute în urma acestui scenariu de testarea am concluzionat că modulul pentru măsurarea umidității solului funcționează în mod adecvat.

### Testarea fotorezistorului

Pentru testarea fotorezistorului am folosit și o rezistență fix iar conectarea acestora la plăcuță R3 UNO + WiFi a fost identică cu modul în care sunt acestea conectate la o plăcuță Arduino UNO în figura 6.10.

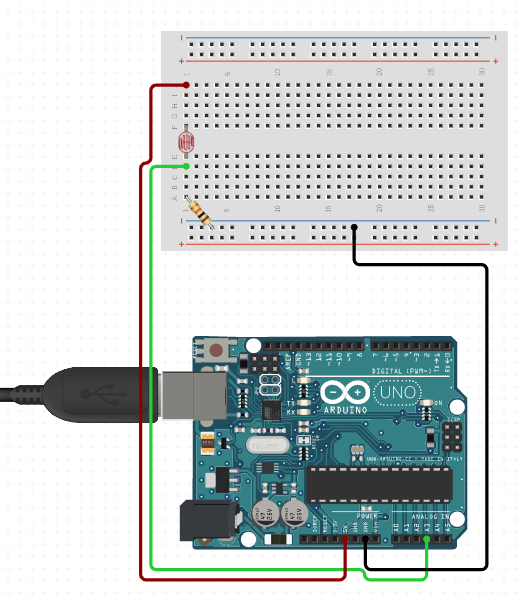


Figura . Schemă petnru conectarea fotorezistorului și rezistenței

Ca în scenariile anterioare de testare a senzorilor, am transmis datele citite serial și am folosit monitorul serial pentru a le vizualiza. Observăm în figura 6.11 stânga valorile înregistrate când fotorezistorul este într-o cameră întunecoasă.

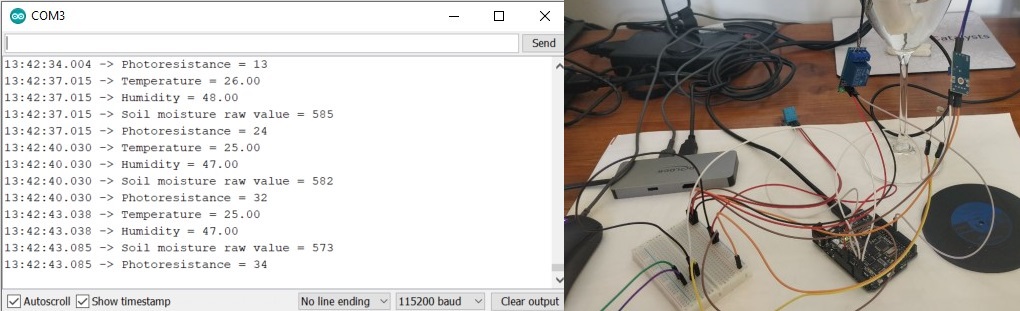


Figura . Fotorezistor și valori înregistrare în semi-întuneric

În figura 6.12 am supus fotorezistorul la o sursă de lumină puternică. Aceasta a crescut conductivitatea fotorezistorului și putem observa cum valoarea înregistrată a crescut și ea.

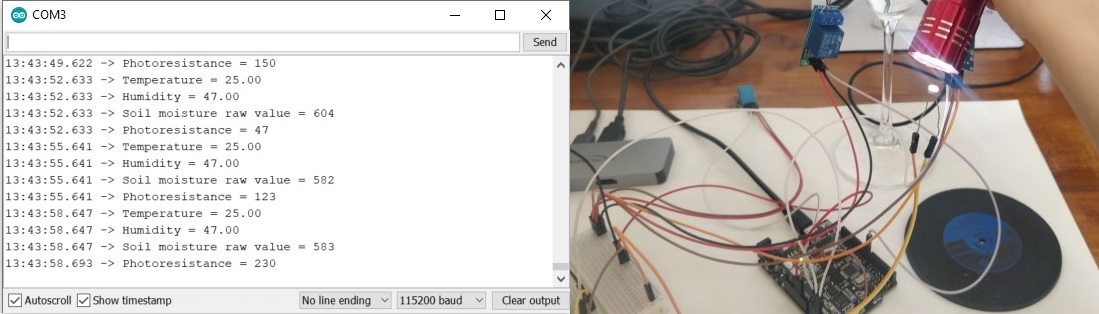


Figura . Fotorezistor și valori înregistrate în lumină puternică

Conform rezultatelor obținute în urma acestui scenariu de testarea am concluzionat că fotorezistorul funcționează în mod adecvat.

### Testare mini pompa de apă

Pentru testarea minipompei am apelat la un releu cu care să putem alimenta pompa de la o altă sursă de tensiune în locul plăcuței, iar plăcuța să fie responsabilă doar de controlul pompei (pornirea/oprirea acesteia). De asemenea ne-am mai folosit de modulul de umiditate a solului testat în prealabil.

Scenariul pe care l-am utilizat a fost: citirea valorii înregistrate de modulul de umiditate a solului iar când valoarea aceasta crește peste 700 (solul devine uscat, deci rezistența dintre sondele senzolui scade) activarea mini pompei submersibile.

Pentru implementarea scenariulului am conectat modulul de măsurare a umidității solului la plăcuță, am conectat pompa la releu, și am legat pinul de intrare date al releului la un pin digital, setat pentru ieșire, al plăcuței. Pompa am plasat-o într-un pahar transparent cu apă ca să observăm că întradevă pompează apa.

În figura 6.13 putem observa ca senzorul este plasat în paharul cu apă, deci înregistrează o conductivitatea mare (rezistență scăzută) astfel popa este oprită.

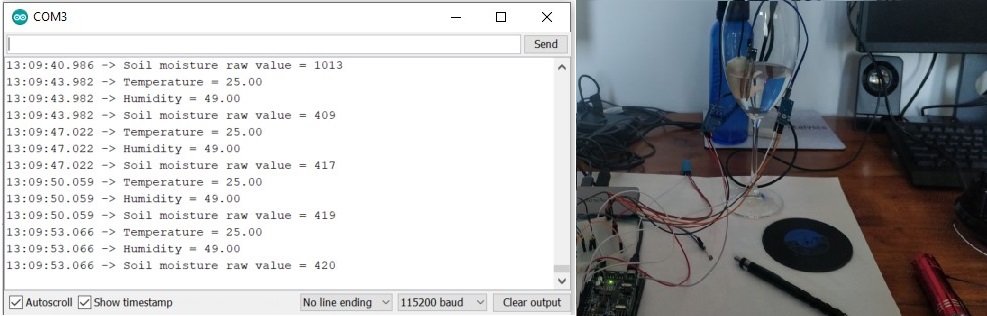


Figura . Valori înregistrate de senzori și mini pompa oprită

În figura 6.14 observăm că am scos senzorul din pahar și l-am lăsat cu sondele în aer, unde acesta înregistrează o rezistență mult mai mare (conductivitatea aerului fiind mult mai mică decât a apei) astfel că mini pompa a început să pomeze apă în pahar.

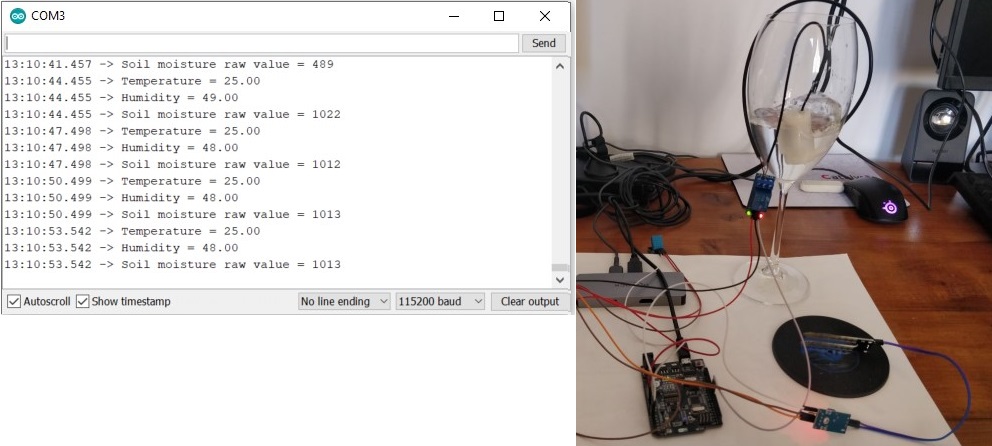


Figura . Valori înregistrate de senzori și mini pompă pornită

Conform rezultatelor obținute în urma acestui scenariu de testarea am concluzionat că mini pompa de apă funcționează în mod adecvat.

# Manual de Instalare si Utilizare

În secţiunea de Instalare trebuie să detaliaţi resursele software şi hardware necesare pentru instalarea şi rularea aplicaţiei, precum şi o descriere pas cu pas a procesului de instalare. Instalarea aplicaţiei trebuie să fie posibilă pe baza a ceea ce se scrie aici.

În acest capitol, trebuie să descrieţi cum se utilizează aplicaţia din punct de vedere al utilizatorului, fără a menţiona aspecte tehnice interne. Folosiţi capturi ale ecranului şi explicaţii pas cu pas ale interacţiunii. Folosind acest manual, o persoană ar trebui să poată utiliza produsul vostru.

# Concluzii

Cca. 5% din total.

Capitolul ar trebui sa conţină (nu se rezumă neapărat la):

* un rezumat al contribuţiilor voastre
* analiză critică a rezultatelor obţinute
* descriere a posibilelor dezvoltări şi îmbunătăţiri ulterioare

# Bibliografie

[1] A. Bak, S. Bouchafa, and D. Aubert, "Detection of independently moving objects through stereo vision and ego-motion extraction," in *IEEE Intelligent Vehicles Symposium (IV)*, San Diego, USA, 2010, pp. 863-870.

[2] A. Chambolle and T. Pock, "A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging," *Journal of Mathematical Imaging and Vision,* vol. 40, pp. 120-145, 2011.

[3] R. C. Gonzalez and R. E. Woods, *Digital Image Processing. Second Edition.*: Addison-Wesley Longman Publishing Co., Inc., 2001.

[4] Ajax Tutorial, <http://www.tutorialspoint.com/ajax/>.

[1] Yuval N. Harari, „*Sapiens : a Brief History of Humankind”,*  New York :Harper, 2015.

[2] Rouse, Margaret (2019). *"internet of things (IoT)".* IOT Agenda. Retrieved 14 August 2019.

[3] Brown, Eric (20 September 2016). *"21 Open Source Projects for IoT".* Linux.com. Retrieved 23 October 2016.

[4] *"Internet of Things Global Standards Initiative".* ITU. Retrieved 26 June 2015.

[5] *"The "Only" Coke Machine on the Internet"*. Carnegie Mellon University. Retrieved 10 November 2014.

[6] fig 1.2 <https://aws.amazon.com/blogs/iot/aws-iot-driven-precision-agriculture/>

[7] fig 1.3 <https://depositphotos.com/184063154/stock-illustration-hand-controlling-kettle-with-smartphone.html>

[8] fig 2.1

[9] Dr. Jennifer S. Raj, J. Vijitha Ananthi, „*AUTOMATION USING IOT IN GREENHOUSE ENVIRONMENT”,* Journal of Information Technology and Digital World, Vol.01/ No. 01, pp. 38-47, 2019

[10] S Aishwarya, R. Thiuzhika, A. Manepally, *„IoT based greenhouse”,* International Journal of Advance Research, Ideas and Innovations in Technology, Volume 4, Issue 5, 2018.

[11] Carența de fier a plantelor <https://www.marcoser.ro/consultanta/boli-daunatori-si-probleme-in-culturile-legumicole/carenta-de-fier/>

[12] N. Thirer, I. Uchansky, *„An FPGA Based Computer System for Greenhouse Control”*, Athens Journal of Sciences, Volume 2, Issue 1, pp. 23-32, 2015.

[13] Shreyas Bhujbal, Yash Deshpande, Arpit Gupta, Ojas Bhelsekar, *„IOT Based Smart Greenhouse”,* in International Journal of Innovative Research in Science, Engineering and Technology*,* Vol. 7, Issue 1, January 2018

[14] Netduino, Netduino 3 WiWi <https://en.wikipedia.org/wiki/Netduino>

[15] Arduino vs Netduino vs Raspberry Pi vs Beaglebone Black <https://www.dragonflythingworks.com/2013/11/arduino-vs-netduino-vs-raspberry-pi-vs-beaglebone-black-part-1/>

[16] Preț de achiziție Arduino Uno <https://cleste.ro/placa-de-dezvoltare-arduino-uno.html>

[17] Preț achiziție Raspberry Pi <https://cleste.ro/raspberry-pi-4-model-b-2gb.html>

[18] Preț de achiziție FPGA <https://cleste.ro/placa-dezvoltare-altera-deo-nano-p0082.html>

[19] Preț de achiziție Uno + Wi-Fi R3 <https://cleste.ro/atelier/placa-dezvoltare-unowifi-r3/>

[20] Preț achiziție Neduino 3 Wi-Fi <https://www.robofun.ro/net/netduino-3-wifi.html>

[21] Preț de achiziție senzor de temperatură și umiditate <https://cleste.ro/senzor-digital-de-temperatura-i-umiditate-dht11-cu-led.html>

[22] Preț de achiziție senzor de umiditate a solului <https://cleste.ro/modul-cu-senzor-umiditate-sol.html>

[23] Preț de achiziție fotorezistor <https://cleste.ro/fotorezistor-5528-ldr.html>

[24] Aplicație de identificare a plantelor „Garden Answers Plant Identification” <https://apps.apple.com/us/app/garden-answers-plant-identification/id605855033>

[25] Aplicație pentru îngrijirea plantelor, „Waterbot: Plants watering + Gardening” <https://play.google.com/store/apps/details?id=net.kosev.watering&hl=en>

[26] Aplicație pentru îngrijirea plantelor, „Vera Plant Care App” <https://www.veraplantcareapp.com>

[27] Robert C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship. Upper Saddle River, NJ: Prentice Hall, 2009.

[28] N.Valliammal, S.N.Geethalakshmi, *„Automatic Recognition System Using Preferential Image Segmentation For Leaf And Flower Images”,* in Computer Science & Engineering: An International Journal (CSEIJ), Vol.1, No.4, pp. 13 – 25, 2011

[30] <https://angular.io/api/core/Component>

[31] <https://cleste.ro/senzor-temperatura-si-umiditate-dht11.html>

[32] <https://cleste.ro/modul-cu-senzor-umiditate-sol.html>

[33] <https://cleste.ro/fotorezistor-5528-ldr.html>

[34] <https://robotdyn.com/uno-wifi-r3-atmega328p-esp8266-32mb-flash-usb-ttl-ch340g-micro-usb.html>

[35] Arduino – <https://www.arduino.cc/en/main/software>

[36] UART - <https://www.circuitbasics.com/basics-uart-communication/#:~:text= UART%20stands%20for%20Universal%20Asynchronous,transmit%20and%20receive%20serial%20data>.

[37] Manager de plăcuță (Board manager) specific ESP8266 http://arduino. esp8266.com/stable/package\_esp8266com\_index.json

[38] Exemplul “Blink” al bibliotecii Arduino <https://www.arduino.cc/en/tutorial/blink>

[39] Tutorial de conectarea a senzorului DHT11 <https://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/>

[40] Bibliotecă pentru utilizare senzorului DHT11 <https://www.circuitbasics.com/wp-content/uploads/2015/10/DHTLib.zip>

[41] Configurarea serverelor mocked în Postman https://learning.postman.com /docs/designing-and-developing-your-api/mocking-data/setting-up-mock/

[45] Git <https://git-scm.com>

# Anexa 1 (dacă este necesar)

…

Secţiuni relevante din cod

…

Alte informaţii relevante (demonstraţii etc.)

…

Lucrări publicate (dacă există)

etc.

1. <https://aws.amazon.com/blogs/iot/aws-iot-driven-precision-agriculture/> [↑](#footnote-ref-2)
2. <https://depositphotos.com/184063154/stock-illustration-hand-controlling-kettle-with-smartphone.html> [↑](#footnote-ref-3)
3. <https://www.marcoser.ro/consultanta/boli-daunatori-si-probleme-in-culturile-legumicole/carenta-de-fier/> [↑](#footnote-ref-4)
4. <https://cleste.ro/raspberry-pi-4-model-b-2gb.html> [↑](#footnote-ref-5)
5. <https://www.dragonflythingworks.com/2013/11/arduino-vs-netduino-vs-raspberry-pi-vs-beaglebone-black-part-1/> [↑](#footnote-ref-6)
6. <https://cleste.ro/placa-de-dezvoltare-arduino-uno.html> [↑](#footnote-ref-7)
7. <https://www.robofun.ro/net/netduino-3-wifi.html> [↑](#footnote-ref-8)
8. <https://www.distrelec.ro/ro/fpga-board-uart-usb-digilent-410-183/p/30044247> [↑](#footnote-ref-9)
9. <https://cleste.ro/atelier/placa-dezvoltare-unowifi-r3/> [↑](#footnote-ref-10)
10. <https://cleste.ro/raspberry-pi-4-model-b-2gb.html> [↑](#footnote-ref-11)
11. <https://cleste.ro/placa-de-dezvoltare-arduino-uno.html> [↑](#footnote-ref-12)
12. <https://www.robofun.ro/net/netduino-3-wifi.html> [↑](#footnote-ref-13)
13. <https://www.distrelec.ro/ro/fpga-board-uart-usb-digilent-410-183/p/30044247> [↑](#footnote-ref-14)
14. <https://cleste.ro/atelier/placa-dezvoltare-unowifi-r3/> [↑](#footnote-ref-15)
15. <https://cleste.ro/senzor-digital-de-temperatura-i-umiditate-dht11-cu-led.html> [↑](#footnote-ref-16)
16. <https://cleste.ro/modul-cu-senzor-umiditate-sol.html> [↑](#footnote-ref-17)
17. <https://cleste.ro/fotorezistor-5528-ldr.html> [↑](#footnote-ref-18)
18. <https://apps.apple.com/us/app/garden-answers-plant-identification/id605855033> [↑](#footnote-ref-19)
19. <https://play.google.com/store/apps/details?id=net.kosev.watering&hl=en> [↑](#footnote-ref-20)
20. <https://www.veraplantcareapp.com> [↑](#footnote-ref-21)
21. <https://apps.apple.com/us/app/garden-answers-plant-identification/id605855033> [↑](#footnote-ref-22)
22. <https://kotlinlang.org> [↑](#footnote-ref-23)
23. <https://gradle.org> [↑](#footnote-ref-24)
24. <https://spring.io/projects/spring-boot> [↑](#footnote-ref-25)
25. <https://start.spring.io> [↑](#footnote-ref-26)
26. <https://en.wikipedia.org/wiki/API> [↑](#footnote-ref-27)
27. <https://swagger.io/specification/> [↑](#footnote-ref-28)
28. <https://openapi-generator.tech> [↑](#footnote-ref-29)
29. <https://www.json.org/json-en.html> [↑](#footnote-ref-30)
30. <https://yaml.org> [↑](#footnote-ref-31)
31. <https://editor.swagger.io> [↑](#footnote-ref-32)
32. <https://openapi-generator.tech/docs/generators/kotlin-spring> [↑](#footnote-ref-33)
33. <https://openapi-generator.tech/docs/installation> [↑](#footnote-ref-34)
34. <https://github.com/OpenAPITools/openapi-generator/tree/master/modules/openapi-generator-gradle-plugin> [↑](#footnote-ref-35)
35. <https://spring.io/projects/spring-data-jpa> [↑](#footnote-ref-36)
36. <https://www.oracle.com/java/technologies/persistence-jsp.html> [↑](#footnote-ref-37)
37. <https://www.postgresql.org> [↑](#footnote-ref-38)
38. <https://www.dbvis.com> [↑](#footnote-ref-39)
39. <https://flywaydb.org> [↑](#footnote-ref-40)
40. <https://angular.io> [↑](#footnote-ref-41)
41. <https://wuschools.com/what-is-mvc-and-understanding-the-mvc-pattern-in-angular/> [↑](#footnote-ref-42)
42. <https://openapi-generator.tech/docs/generators/typescript-angular> [↑](#footnote-ref-43)
43. <https://www.npmjs.com/package/npm> [↑](#footnote-ref-44)
44. <https://verdaccio.org> [↑](#footnote-ref-45)
45. <https://www.arduino.cc/en/main/software> [↑](#footnote-ref-46)
46. <https://www.adafruit.com/product/386> [↑](#footnote-ref-47)
47. <https://cleste.ro/senzor-temperatura-si-umiditate-dht11.html> [↑](#footnote-ref-48)
48. <https://www.circuitstoday.com/arduino-soil-moisture-sensor> [↑](#footnote-ref-49)
49. <https://cleste.ro/modul-cu-senzor-umiditate-sol.html> [↑](#footnote-ref-50)
50. <https://en.wikipedia.org/wiki/Photoresistor> [↑](#footnote-ref-51)
51. <https://cleste.ro/fotorezistor-5528-ldr.html> [↑](#footnote-ref-52)
52. <https://cleste.ro/pompa-de-apa-3-6v.html> [↑](#footnote-ref-53)
53. <https://cleste.ro/modul-releu-1-canal-5v.html> [↑](#footnote-ref-54)
54. http://arduino. esp8266.com/stable/package\_esp8266com\_index.json [↑](#footnote-ref-55)
55. <https://www.circuitbasics.com/basics-uart-communication> [↑](#footnote-ref-56)
56. <https://git-scm.com> [↑](#footnote-ref-57)
57. <https://www.arduino.cc/en/tutorial/blink> [↑](#footnote-ref-58)
58. http://arduino. esp8266.com/stable/package\_esp8266com\_index.json [↑](#footnote-ref-59)
59. https://learning.postman.com /docs/designing-and-developing-your-api/mocking-data/setting-up-mock/ [↑](#footnote-ref-60)
60. <https://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/> [↑](#footnote-ref-61)
61. https://www.circuitbasics.com/wp-content/uploads/2015/10/DHTLib.zip [↑](#footnote-ref-62)